



インターシステムズの自然言語処理 (NLP) の使用法

Version 2023.1
2024-01-02

インターシステムズの自然言語処理 (NLP) の使用法

InterSystems IRIS Data Platform Version 2023.1 2024-01-02

Copyright © 2024 InterSystems Corporation

All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, および TrakCare は、InterSystems Corporation の登録商標です。HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, および InterSystems TotalView™ For Asset Management は、InterSystems Corporation の商標です。TrakCare は、オーストラリアおよび EU における登録商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を移動および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

目次

1 コンセプトの概要	1
1.1 シンプルな使用事例	1
1.2 NLP とは何か	2
1.2.1 NLP が提供しない機能	3
1.3 NLP が識別する論理テキスト・ユニット	3
1.3.1 文	3
1.3.2 エンティティ	4
1.3.3 CRC と CC	5
1.3.4 パス	5
1.4 スマート・インデックス作成	5
1.5 スマート・マッチング	6
2 NLP の実装	7
2.1 サンプル・プログラムに関するメモ	7
2.2 %Persistent オブジェクト・メソッドに関するメモ	8
2.3 %iKnow と %SYSTEM.iKnow に関するメモ	8
2.4 必要なディスク容量と NLP グローバル	8
2.4.1 バッチ・ロードのスペース割り当て	9
2.5 入力データ	9
2.5.1 ファイル形式	9
2.5.2 SQL レコード形式	10
2.5.3 テキストの正規化	10
2.5.4 ユーザ定義のソースの正規化	10
2.6 出力構造	11
2.7 定数	11
2.8 エラー・コード	11
3 ドメイン・アーキテクト	13
3.1 アーキテクトへのアクセス	13
3.1.1 ネームスペースの有効化	13
3.2 ドメインの作成	14
3.2.1 ドメインを開く方法	15
3.2.2 ドメイン定義の変更	15
3.2.3 ドメインの削除	15
3.3 モデル要素	15
3.3.1 ドメイン設定	16
3.3.2 メタデータ・フィールド	16
3.3.3 データ位置	17
3.3.4 Skiplist	20
3.3.5 マッチング	20
3.4 保存、コンパイル、およびビルド	21
3.5 ドメイン・エクスプローラ	21
3.5.1 ドメイン・エクスプローラの設定	22
3.5.2 すべての概念のリスト	23
3.5.3 指定したエンティティの分析	23
3.5.4 分析するソースの制限	25
3.6 インデックス作成結果	26
3.6.1 ドメイン・データ	26

3.6.2 手動入力データ	26
3.6.3 [インデックス付きの文]	27
3.6.4 [概念] および [CRC]	27
4 REST インタフェース	29
4.1 Swagger	29
4.1.1 Swagger を使用して NLP データを返す	30
4.2 REST 操作	30
4.2.1 ドメインと構成	31
4.2.2 ソース	31
4.2.3 エンティティ	32
4.2.4 文	33
4.2.5 パスと CRC	33
4.2.6 デictionaryとマッチング	34
4.2.7 Skiplist	35
5 手動による NLP 環境の作成	37
5.1 NLP ドメイン	37
5.1.1 DomainDefinition を使用したドメインの定義	38
5.1.2 プログラムによるドメインの定義	40
5.1.3 ドメイン・パラメータの設定	40
5.1.4 ドメインへの割り当て	43
5.1.5 ドメインからの全データの削除	43
5.1.6 全ドメインのリスト	44
5.1.7 ドメイン名の変更	45
5.1.8 ドメインのコピー	45
5.2 InterSystems NLP 構成	46
5.2.1 構成の定義	46
5.2.2 構成プロパティの設定	46
5.2.3 構成の使用	47
5.2.4 全構成のリスト	48
5.2.5 文字列を正規化する構成の使用法	48
5.3 InterSystems NLP ユーザ・dictionary	49
5.3.1 ドメイン・アーキテクトでのユーザ・dictionaryの定義	50
5.3.2 オブジェクト・インスタンスとしてのユーザ・dictionaryの定義	50
5.3.3 ファイルとしてのユーザ・dictionaryの定義	51
6 プログラムによるテキスト・データのロード	53
6.1 ローダ	53
6.1.1 ローダ・エラーのログ	54
6.1.2 ローダの Reset()	54
6.2 リスタ	54
6.2.1 リスタの初期化	54
6.2.2 リスタ・インスタンス既定のオーバーライド	55
6.2.3 リスタによるソースへの ID の割り当て	56
6.2.4 リスタの既定値の例	56
6.2.5 リスタ・パラメータ	57
6.2.6 バッチとリスト	57
6.3 リストとロードの例	58
6.3.1 ファイルのロード	58
6.3.2 SQL レコードのロード	58
6.3.3 添え字付きグローバルの要素のロード	59

6.3.4 文字列のロード	60
6.4 ドメイン・コンテンツの更新	61
6.4.1 ソースの追加	61
6.4.2 ソースの削除	62
6.5 仮想ソースのロード	62
6.5.1 仮想ソースの削除	63
6.6 ロードされたソース・データのコピーとインデックス再作成	63
6.6.1 UserDictionary とコピーされたソース	65
7 テキストのロード時のパフォーマンスに関する考慮事項	67
8 NLP クエリ	69
8.1 クエリのタイプ	69
8.2 この章で説明するクエリ	69
8.3 クエリ・メソッド・パラメータ	70
8.4 ソースと文のカウンタ	71
8.5 エンティティのカウンタ	72
8.6 上位エンティティのリスト	73
8.6.1 GetTop() : 最も頻繁に出現しているエンティティ	73
8.6.2 GetTopTFIDF() および GetTopBM25()	74
8.7 CRC クエリ	76
8.7.1 エンティティを含む CRC のリスト	76
8.7.2 CRC を含むソースのカウンタ	77
8.7.3 CRC マスクを満たす 1 つまたは複数の文のリスト	78
8.8 類似エンティティのリスト	79
8.8.1 パーツと N-gram	80
8.9 関連するエンティティのリスト	80
8.9.1 ポジションによる制限	81
8.10 パスのカウンタ	83
8.11 類似ソースのリスト	83
8.12 ソースの要約	85
8.12.1 カスタム・サマリ	87
8.13 ソースのサブセットのクエリ	89
9 意味的属性	91
9.1 属性が機能するしくみ : マーカ用語と属性拡張	91
9.1.1 エンティティ・レベル : マーカ用語のビット・マスク	91
9.1.2 パスレベル : 拡張属性の位置と範囲	92
9.2 属性データへのアクセス	92
9.2.1 属性のデータ構造	92
9.2.2 例	93
9.3 サポートされる属性	94
9.4 否定	94
9.4.1 否定の特殊なケース	95
9.4.2 否定とスマート・マッチング	95
9.5 時間、期間、および頻度	96
9.6 測定	96
9.7 感情	97
9.8 確実性	97
9.9 汎用属性	98
10 語幹解析	99
10.1 語幹解析の構成	99

10.1.1 Hunspell	100
10.2 語幹取得のメソッド	100
10.3 語幹の使用	100
11 Skiplist	101
11.1 skiplist の作成	101
11.1.1 skiplist とドメイン	102
11.2 skiplist をサポートするクエリ	103
11.2.1 skiplist クエリの例	104
12 ソースのフィルタ処理	107
12.1 サポートされるフィルタ	107
12.2 ソースの ID によるフィルタ処理	108
12.2.1 外部 ID でフィルタ処理	108
12.2.2 ソース ID でフィルタ処理	109
12.3 フィルタ処理によるソースからのランダムな選択	109
12.4 文の数によるフィルタ処理	111
12.5 エンティティー一致によるフィルタ処理	112
12.5.1 ディクショナリー一致によるフィルタ処理	113
12.6 インデックス作成日メタデータによるフィルタ処理	114
12.7 ユーザ定義メタデータによるフィルタ処理	115
12.7.1 メタデータ・フィルタ演算子	116
12.8 SQL クエリによるフィルタ	117
12.9 フィルタ・モード	118
12.10 GroupFilter の使用による複数フィルタの組み合わせ	118
13 テキスト・カテゴリ化	121
13.1 テキスト・カテゴリ化の実装	121
13.1.1 実装インタフェース	122
13.1.2 テキスト分類モデルの管理	122
13.2 トレーニング・セットとテスト・セットの確立	123
13.3 プログラムによるテキスト分類子の構築	124
13.3.1 テキスト分類子の作成	124
13.3.2 用語ディクショナリの生成	125
13.3.3 分類オプティマイザの実行	126
13.3.4 テキスト分類子の生成	126
13.4 テキスト分類子のテスト	127
13.4.1 テスト結果の使用	128
13.5 UI を使用したテキスト分類子の構築	128
13.5.1 UI に対するデータ・セットの定義	128
13.5.2 テキスト分類子の構築	129
13.5.3 テキスト分類子の最適化	130
13.5.4 データのテスト・セットに対するテキスト分類子のテスト	131
13.5.5 未分類データに対するテキスト分類子のテスト	131
13.6 テキスト分類子の使用	131
14 優位性と近似	133
14.1 意味的優位性	133
14.1.1 コンテキストにおける優位性	134
14.1.2 意味的優位性の概念	134
14.1.3 意味的優位性の例	135
14.2 語義的な近似	136
14.2.1 日本語の語義的な近似	137

14.2.2 近似の例	137
15 カスタム・メトリック	139
15.1 カスタム・メトリックの実装	139
15.2 タイプおよびターゲット	140
15.3 メトリックのコピー	141
16 スマート・マッチング：ディクショナリの作成	143
16.1 ディクショナリ構造とマッチングの概要	143
16.1.1 用語	144
16.2 ディクショナリの作成	144
16.2.1 ディクショナリおよびドメイン	145
16.2.2 ディクショナリ作成例	146
16.2.3 形式用語の定義	147
16.2.4 ディクショナリ用語における複数の形式	148
16.3 ディクショナリのリストおよびコピー	148
16.3.1 既存のディクショナリのリスト	148
16.3.2 ディクショナリのコピー	149
16.4 ディクショナリ構造の拡張	149
17 スマート・マッチング：ディクショナリの使用	151
17.1 ディクショナリ・マッチングの仕組み	151
17.1.1 一致スコア	151
17.2 文字列のマッチング	152
17.2.1 エンティティ文字列のマッチング	152
17.2.2 文文字列のマッチング	154
17.3 ソースのマッチング	156
17.4 マッチング・プロファイルの定義	158
17.4.1 マッチング・プロファイル・プロパティ	159
17.4.2 ドメインの既定マッチング・プロファイル	159
18 ユーザ・インタフェース	163
18.1 NLP ユーザ・インタフェースの表示方法	163
18.2 抽象ポータル	163
18.3 抽象ソース・ビューワ	164
18.4 ロード・ウィザード	164
18.5 ドメイン・エクスプローラ	164
18.6 基本ポータル	164
18.7 インデックス作成結果	165
18.8 マッチング結果	165
19 InterSystems IRIS 自然言語処理 (NLP) ツール	167
19.1 NLP シェル・インタフェース	167
19.1.1 ソースのリスト、表示、および要約	167
19.1.2 ソースのフィルタ処理	169
19.2 NLP データ・アップグレード・ユーティリティ	169
20 iKnow Web サービス	171
20.1 利用可能な Web サービス	171
20.2 NLP Web サービスの使用法	172
20.3 例	172
20.4 NLP Web サービスと主要な NLP API の比較	173
20.5 関連項目	174
21 KPI とダッシュボード	175

21.1 KPI 用語	175
21.2 テキスト分析クエリを使用する KPI の定義	176
21.3 利用可能な KPI フィルタ	177
21.4 KPI プロパティのオーバーライド	177
21.5 例	178
21.6 KPI を表示するダッシュボードの作成	178
21.6.1 ダッシュボードの作成 : Basics	179
21.6.2 系列名の変更	180
21.6.3 プロパティの構成	181
21.6.4 [前のページ] および [次のページ] ボタンの追加	182
21.6.5 最大表示長の設定	183
21.6.6 KPI でのダッシュボードの例	183
21.7 ダッシュボードへのアクセス	184
21.8 関連項目	184
22 NLP のカスタマイズ	185
22.1 カスタム・リスタ	185
22.1.1 リスタ名	186
22.1.2 SplitFullRef() と BuildFullRef()	186
22.1.3 既定のプロセッサ	186
22.1.4 リストの展開	186
22.2 カスタム・プロセッサ	187
22.2.1 メタデータ	187
22.3 カスタム・コンバータ	187
22.3.1 %OnNew	187
22.3.2 BufferString	188
22.3.3 Convert	188
22.3.4 NextConvertedPart	188
23 言語の識別	189
23.1 自動言語識別の構成	189
23.2 自動言語識別の使用	190
23.2.1 言語識別クエリ	190
23.3 自動言語識別のオーバーライド	192
23.4 言語固有の問題	192
付録A: ドメイン・パラメータ	193

テーブル一覧

テーブル I-1: Basic ドメイン・パラメータ	194
テーブル I-2: Advanced ドメイン・パラメータ	198

1

コンセプトの概要

NLP 意味分析エンジンは、InterSystems IRIS® Data Platform 上の構造化されていないデータ、つまり、英語やフランス語などの人間の言語でテキストとして記述されたデータの分析に使用されます。NLP はこの種のデータにすばやくアクセスして分析することができるため、ユーザはすべてのデータを取り扱うことが可能になります。NLP では、このデータのコンテンツに関する予備知識は一切必要とされません。また、NLP がサポートする言語の 1 つで記述されている限り、どの言語で記述されているかを知っておく必要もありません。

構造化されていないデータは一般的に、複数のソース・テキスト（多くの場合は非常に多数のテキスト）から構成されます。ソース・テキストは通常、句読点によって文に分割されたテキストから構成されます。ファイル、SQL 結果セットのレコード、一連のブログ・エントリといった Web ソースなどがソース・テキストになります。

1.1 シンプルな使用事例

以下の文を例にとり、構造化されていないデータを NLP がどのように処理するかを見てみましょう。

General Motors builds their Chevrolet Volt in the Detroit-Hamtramck assembly plant.

まず、NLP によってテキストが文に分割されます。次に、各文の言語が自動的に特定され、その言語（言語モデル）に対応する一連の意味規則に基づいて各文が意味的に分析されます。文中の単語を“理解”したり調べたりする必要はありません。この文の分析で得られた意味的エンティティに NLP によってインデックスが作成されます（文字は小文字に正規化されます）。

[general motors] [builds] (their) [chevrolet volt] [in] the [detroit-hamtramck assembly plant]

NLP は、最初の文から以下のエンティティを識別しています。

- ・ 3 つのコンセプト：[general motors] [chevrolet volt] [detroit-hamtramck assembly plant]
- ・ 2 つのリレーション：{builds} {in}
- ・ 1 つのパス関係：(their)（パス関係はパス分析で考慮されますが、インデックスは作成されません。）
- ・ 1 つの無関係：the（無関係なエンティティは破棄されて、以降の NLP 分析からは除外されます。）

注釈 この例では図示するために、各コンセプトを角括弧で、各リレーションを中括弧で、各パス関係を丸括弧で区切っていますが、NLP 内ではこのような区切り文字は使用されません。

NLP は、エンティティごとに一意の ID を割り当てます。また、NLP は、コンセプト - リレーション - コンセプト (CRC) というパターンに従うエンティティのシーケンスを識別します。この例では、以下の 2 つの CRC が識別されています。

[general motors] {builds} [chevrolet volt]
[chevrolet volt] {in} (their) [detroit-hamtramck assembly plant]

NLP は、CRC ごとに一意の ID を割り当てます。

また、NLP は、この文に連続するエンティティのシーケンス(ここでは、シーケンス CRCRC)が含まれていることを認識します。全部で 1 つのステートメントを表す、文内のエンティティのシーケンスをパスと呼びます。文全体が 1 つのパスになることも、文全体に複数のパスが含まれていることもあります。NLP は、パスごとに一意の ID を割り当てます。

これで、NLP はテキストに含まれる文、各文の関連するエンティティ、どのエンティティがコンセプトでどのエンティティがリレーションか、どのエンティティが CRC のシーケンスを形成するか、また、どのエンティティのシーケンスがパスを形成するかを識別しました。NLP はこれらの意味的ユニット(意味的単位)を使用することで、ソース・テキストのコンテンツに関するさまざまなタイプの意味ある情報を返すことができます。

1.2 NLP とは何か

NLP は、関連付けられたリレーショナル・エンティティにテキストを分割し、これらのエンティティのインデックスを生成することで、構造化されていないデータへのアクセスを提供します。これはテキストを文に分割し、次にそれぞれの文をコンセプトやリレーションのシーケンスに分割します。NLP は、テキストの言語(英語など)を識別してから、対応する NLP 言語モデルを適用することで、この処理を実行します。

- ・ リレーションとは、2 つのコンセプトの関係を指定することでそれらのコンセプトを結合する単語や単語のグループです。NLP には、文のリレーションを識別できるコンパクトな言語モデルが含まれています。
- ・ コンセプトとは、リレーションによって関連付けられる単語や単語のグループです。NLP は、リレーションを判別することで、関連付けられるコンセプトを識別できます。このようにして、NLP 分析エンジンは、コンセプトのコンテンツを“理解”しなくても、コンセプトを意味的に識別できます。

注釈 説明するならば、動詞は一般的にはリレーションになり、形容詞が関連付けられた名詞は一般的にはコンセプトになります。ただし、リレーションとコンセプトの言語モデルは、単なる動詞と名詞の区別よりはるかに高度で包括的です。

このように、NLP はコンセプト(C)とリレーション(R)に文を分割します。言語モデルは、リレーションを識別するための一連のコンテキスト・ルールと関係用の単語を含む比較的小さな固定ディクショナリを使用します。リレーションと識別されないものは、すべてコンセプトと見なされます(NLP は“the”や“a”のような無関係な単語も識別して、以降の分析からは除外します)。

リレーションとコンセプトは総称してエンティティと呼ばれます。ただし、関連付けられたコンセプトなしでリレーションが意味を持つことはほとんどありません。このため、NLP エンティティ分析では、リレーションによって関連付けられたコンセプトを含むシーケンスおよびコンセプトに重点を当てます。

NLP はリレーションの識別に焦点を当てた小さく安定した言語モデルを使用してテキストを分析するため、どのような内容を含むテキストのインデックスも迅速に作成できます。NLP はコンセプトを識別するためにディクショナリやオントロジを使用する必要はありません。

NLP が 1 つまたは(より一般的には)多数のテキストの各文のコンセプトとリレーションを識別すると、この情報を使用して以下のタイプの処理を実行できます。

- ・ **スマート・インデックス作成** : 構造化されていない大量のテキスト本文から、何が関係性があり、何が関連し、何が代表的かに対する洞察を提供します。
- ・ **スマート・マッチング** : ソース・テキスト内のエンティティを、リストやディクショナリといった外部項目と関連付けるための方法を提供します。これらのリストには、完全一致(同一)と部分一致のための単語、語句、または文を含むことができ、形式によるマッチングのためのテンプレートを含むことができます。

1.2.1 NLP が提供しない機能

NLP は検索ツールではありません。検索ツールでは、テキスト内にあると既に確信しているものだけを見つけることができます。インターシステムズは、SQL テーブルにある未構造化テキスト・データの検索ツールとして [InterSystems SQL Search](#) を提供しています。InterSystems SQL Search では、NLP の機能を多く使用することにより、インテリジェントなテキスト検索を実現しています。

NLP とは、コンテンツ分析ツールのことです。NLP を使用すると、コンテンツがまったく未知のテキストを含めて、テキスト・データのコンテンツ全体を使用できます。

NLP はディクショナリに基づいたコンテンツ・アナライザではありません。ディクショナリに基づいたツールとは異なり、個々の単語に文を区切ってからこれらの単語を“理解”してコンテキストの再構成を試みることはありません。NLP は単純にエンティティを意味的に識別します。iKnow がこれらのエンティティをディクショナリやオントロジで調べる必要はありません。このため、その言語モデルはコンパクトで安定し、多目的に使用できるものとなっており、分析するテキストの種類（医療分野や法務分野など）に関する情報を指定したり、関連用語の別個のディクショナリを提供する必要はありません。NLP は、用語のディクショナリやオントロジを関連付けることで拡張できますが、基本的な機能ではこれらは必要とされません。したがって、iKnow では、ディクショナリの作成、カスタマイズ、または定期的な更新は必要とされません。

NLP は、語幹解析をサポートしていますが、既定では語幹解析ツールではありません。既定では、語幹形式へのリレーションやコンセプトの縮小は行いません。その代わりに、各要素を別個のエンティティとして取り扱って、他の要素に対する類似性の程度を識別します。NLP は、オプション機能として [語幹解析](#) をサポートします。この機能は、主にロシア語とウクライナ語のテキスト・ソースに使用することをお勧めします。“インターシステムズ・クラス・リファレンス”で説明されているように、InterSystems IRIS は、**%Text** パッケージという語幹解析の実行のために使用できる一連のクラスも提供します。**%Text** と **%iKnow** は、互いに完全に独立しており、異なる目的に使用されます。

1.3 NLP が識別する論理テキスト・ユニット

1.3.1 文

NLP は言語モデルを使用して、ソース・テキストを文に分割します。NLP では通常、文ターミネータ（通常は句読点記号）とそれに続く最低 1 つの空白または改行により終わるテキスト単位として文を定義します。次の文は、次の非空白文字で始まります。文の開始を示すための大文字と小文字の区別は不要です。

ほとんどの言語において、文ターミネータには、ピリオド (.)、疑問符 (?)、感嘆符 (!)、およびセミコロン (;) が使用されます。また、省略記号 (...) または強調句読点 (??? または !!!) など、ターミネータを複数使用することもできます。任意のターミネータの組み合わせ (...!?) も使用できます。文ターミネータ間の空白は新しい文を示すので、省略記号に空白が含まれていると (...), 実際には 3 つの文になります。文ターミネータの次には、空白文字（スペース、タブ、または改行）が続くか、または一重引用符か二重引用符に空白文字が続くかのいずれかとなります。例えば、`"Why?" he asked.` は 2 つの文ですが、`"Why?", he asked.` は 1 つの文となります。

2 行改行は、文ターミネータ文字の有無とは関係なく、文ターミネータとして機能します。したがって、タイトルやセクション見出しは、空白行が続いた場合、個別の文と見なされます。また、ファイルの終わりも文ターミネータとして扱われます。したがって、ソースに 1 つでもコンテンツ（空白を除く）が含まれている場合、文ターミネータの存在とは無関係に、少なくとも 1 つの文が含まれることになります。同様に、ファイルの最後のテキストは、文ターミネータの存在とは関係なく、個別の文として扱われます。

通常、ピリオド (.) に空白が続く場合は文の区切りを示しますが、NLP 言語モデルはこの規則の例外を認識します。例えば、英語モデルでは“Dr.” や“Mr.”などの一般的な省略形が認識され（大文字と小文字の区別なし）、文区切りを実行するのではなく、ピリオドを削除します。英語モデルでは、“No.” は省略形として認識されますが、小文字の“no.” は文ターミネータとして扱われます。

構成の [UserDictionary](#) オプションを使用して、特定の場合に文を終了させたり、文の終了を回避することができます。例えば、“Fr.” という Father または Friar の省略形は、英語モデルでは認識されません。これは文の区切りとして処理されます。UserDictionary を使用すると、ピリオドを削除するか、またはこのピリオドの使い方では文区切りが起こらないように指定できます。UserDictionary はソースがロードされるときに適用されます。既にロードされているソースは影響を受けません。

1.3.2 エンティティ

エンティティとは、テキストの最小限の論理ユニット(論理単位)です。エンティティは、単語または単語のグループです。単語のグループとは、1 つの概念または関係に NLP が単語を論理的にまとめたものです。電話番号や電子メール・アドレスなどの他の論理ユニットも、エンティティと見なされます(そして概念として扱われます)。

注釈 日本語のテキストは、コンセプトとリレーションに分別できません。その代わりに、NLP は、助詞の付いたエンティティのシーケンスとして、日本語のテキストを解析します。日本語の“エンティティ”の定義は、その他の NLP 言語におけるコンセプトとほぼ同じです。NLP 日本語サポート(日本語で記述)の説明については、“[NLP Japanese の概要](#)”を参照してください。

NLP は比較やカウントを行うことができるように、エンティティを正規化します。また、無関係な単語を除去します。そしてエンティティを小文字に変換します。さらに、ほとんどの句読点や一部の特殊文字をエンティティから除去します。

既定では、NLP はエンティティの分析をコンセプトに制限します。既定では、リレーションは、コンセプトを連結するという役割によってのみ分析されます。この既定はオーバーライドすることが可能です。詳細は“[NLP クエリ](#)”の章の“[ポジションによる制限](#)”のセクションを参照してください。

1.3.2.1 パス関係の単語

NLP は各言語で特定の単語を文およびパスの分析における主要部分として識別しますが、それ以外は無関係と見なします。文またはパスのコンテキスト以外では、これらの単語には情報を提供する内容がほとんど含まれていません。以下は、一般的なパス関係の単語です。

- ・ あらゆるタイプの代名詞：定代名詞、不定代名詞、所有代名詞。
- ・ 時間、頻度、または場所に関する曖昧な表現。例えば、“then”、“soon”、“later”、“sometimes”、“all”、“here”。

パス関係の単語はコンセプトと見なされず、頻度または優位性の計算でもカウントされません。パス関係の単語は否定マーカまたは時刻属性マーカである場合もあります。例えば、“none”、“nothing”、“nowhere”、“nobody”などです。パス関係の単語は語幹抽出されません。

1.3.2.2 無関係な単語

NLP は、各言語の特定の単語を無関係な単語として識別し、これらの単語を NLP のインデックス作成から除外します。さまざまな種類の無関係な単語があります。

- ・ NLP 言語モデルによって意味的な重要性がほとんどまたはまったくないと見なされる冠詞(“the” や “a” など)やその他の単語。
- ・ “And”、“Nevertheless”、“However”、“On the other hand” など、文頭の前置きの単語や語句。
- ・ スペースまたは文の句読点で分割されていない 150 文字を超える文字列。この長さの“単語”は、テキスト・エンティティではない可能性が非常に高く、したがって NLP のインデックス作成から除外されます。まれな状況(化学の専門用語や URL 文字列など)ではこれらの 150 文字を超える単語が意味を持つため、NLP はこれらに属性“nonsemantic”というフラグを付けます。

無関係な単語は、NLP のインデックス作成から除外されますが、文が表示されるときには保持されます。

1.3.3 CRC と CC

NLP はコンセプト (C) とリレーション (R) に文を分割すると、こうした基本エンティティ間の「つながり」のタイプを判別できます。

- ・ CRC は、コンセプト - リレーション - コンセプトのシーケンスです。CRC は、ヘッド・コンセプト - リレーション - qConcept のシーケンスとして扱われます。エンティティが、ヘッド、リレーション、またはテールのいずれであるかは、ポジションと呼ばれます。場合によっては、CRC のシーケンス・メンバ (CR または RC) の 1 つの文字列値が空になることもあります。これは、例えば、“Robert slept” など、CRC のリレーションが自動詞の場合に起こります。
- ・ CC は、「コンセプト + コンセプト」のペアです。NLP は各コンセプトのポジションを保持しますが、2 つのコンセプトのリレーションは無視します。CC は、2 つの関連付けられたコンセプトとして扱うことも、ヘッド・コンセプト/テール・コンセプトのシーケンスとして扱うことも可能です。CC ペアを使用して、ヘッド/テールのポジションやリンクするリレーションにかかわらず、関連付けられたコンセプトを識別することができます。これはコンセプトのネットワーク、つまり、どのコンセプトが他のどのコンセプトとつながっているかを判断する際に特に役立ちます。また、CC ペアはヘッド/テールのシーケンスとしても使用できます。

注釈 日本語は、CRC や CC に関して意味的に分析することはできません。これは、NLP が日本語のエンティティをコンセプトとリレーションに分別しないためです。

1.3.4 パス

パスは、1 つの文におけるエンティティの意味のあるシーケンスです。西洋語では、一般に、パスは連続する CRC に基づいています。そのため、結果としてのパスには、原文の順序でエンティティ (コンセプトとリレーション) が含まれます。通常、これは、連続する CRC のシーケンスを形成します (例外もあります)。例えば、一般的なパスのシーケンスでは、1 つの CRC のテール・コンセプトが、次の CRC のヘッド・コンセプトになります。この結果、C-R-C-R-C という 5 つのエンティティから成るパスが形成されます。コンセプトとリレーションの他の意味のあるシーケンスもパスとして扱われます。例えば、コンセプトの代役としてパス関係の代名詞を含むシーケンスなどです。

日本語の場合、パスは原文のエンティティのシーケンスに基づくことができません。それでも、NLP は、日本語テキストで意味を持つエンティティのシーケンスとしてパスを識別します。日本語の NLP 意味分析には、エンティティ・ベクトルを作成するためのエンティティ・ベクトル・アルゴリズムが使用されます。NLP が日本語の文をエンティティ・ベクトルに変換するとき、どのエンティティが互いに結び付いているのかとその結び付きの強さを示すために、通常は原文と異なる順序でエンティティをリストします。その結果としてのエンティティ・ベクトルが、パスの解析に使用されます。

パスは少なくとも 2 つのエンティティを含んでいる必要があります。すべての文がパスであるわけではありません。非常に短い文は、パスと見なされるために必要な最小数のエンティティを含まない場合があります。

パスは常に 1 つの文の中に収まります。ただし、1 つの文に複数のパスが含まれる場合があります。これは、連続しないシーケンスが文中に識別された場合に起こる場合があります。パス・シーケンスを含むエンティティは、識別されると、区分けと正規化が施され、パスには一意の ID が割り当てられます。意味を持って関連付けられた一定エンティティを識別できるほど CRC のみの分析が大規模でない場合は、パスが役立ちます。特に、幅広いコンテキストの小さな言語単位が返される場合には、パスは有用となります。

1.4 スマート・インデックス作成

スマート・インデックス作成とは、構造化されていないテキストを、コンセプトとリレーションのリレーショナル・ネットワーク (関係網) に変換するプロセスです。構造化されていない複数のテキストのコンテンツにインデックスを作成し、頻度の順に概念をリストするなど、結果として作成されたインデックス付きのエンティティをユーザが定義したクエリ条件に従って分析できます。それぞれのインデックス付きエンティティは、そのソース・テキスト、ソース文、およびリレーショナル・エンティティ (CRC シーケンスにおけるポジションなど) を参照できます。スマート・インデックス作成の一部として、NLP はインデッ

クスが作成されたそれぞれの概念に、テキストにおけるその概念の合計出現数（頻度）と、その概念が出現するテキストの数（分散）を示す 2 つの値を割り当てます。

複数のテキストについてスマート・インデックス作成を実行すると、NLP はこの情報を使用してソース・テキストを分析できます。例えば、NLP はインテリジェント・コンテンツ参照を実行できます。NLP がインデックスを作成した任意の選択項目から、項目の類似性の程度に基づいて、他の項目を参照できます。インテリジェント参照は、1 つのソース・テキスト内またはインデックスが作成された全ソース・テキスト上で実行できます。

テキストのインデックスを作成すると、NLP は個々のテキストの要約を生成できます。ユーザは元のテキストのパーセンテージとして要約の長さを指定します。NLP は、インデックス統計に基づいて、全体に最も関連性がある元のテキストの文から構成される要約テキストを返します。例えば、100 個の文から構成されるテキストに対して 50% の要約を指定した場合、NLP は最も関連性がある 50 個の文から構成される要約テキストを元のテキストから生成します。

1.5 スマート・マッチング

NLP がテキストのコレクションにインデックスを作成すると、ユーザが定義した 1 つ以上の一致リストに対してテキスト内の項目をマッチングさせ、一致した項目にタグを付けることができます。スマート・マッチングは概念と語句に対して、完全なコンテキストの意味的理解に基づいた高精度のタグ付けを実行します。一致は、類似する概念や語句の間で発生する場合と、完全一致（同一）の場合があります。このタグ付けは意味的一致の検出に基づくため、スマート・マッチングではテキスト・コンテンツを理解する必要は一切ありません。

テキスト内の一致する語句の各出現箇所がタグ付けされると、タグ・テキストとの関連付けが維持されます。これらの語句は、1 つのエンティティ、CRC、またはパスとしてマッチングできます。例えば、国名のリストを指定して、テキストにおける国名の各出現箇所にタグ付けし、迅速なアクセスを実現できます。アナリスト・レポートとマッチングするための会社名のディクショナリを作成することで、関心のある会社の最新ニュースを迅速に見つけることが可能になります。また、指定の医療処置の各出現箇所（多様な言い回しで出現）を医療診断コードにマッチングさせるためのディクショナリを作成することもできます。

ディクショナリ・マッチングは単純なエンティティだけに制限されず、ディクショナリの用語自体が 1 つのエンティティの枠を超える場合は、CRC やパスに拡張します。NLP はソース・テキストにインデックスを作成するのと同じようにディクショナリ用語にインデックスを作成するため、ディクショナリ・エントリは 1 つの文と同じ長さになる可能性があります。類似する情報を見つけるには、ソースに対してディクショナリ・エントリの文をマッチングさせるのが便利な場合もあります。

2

NLP の実装

NLP 意味分析エンジンは、InterSystems IRIS® Data Platform の完全に統合されたコンポーネントです。別途インストールする必要はありません。構成の変更も不要です。

NLP を使用できるかどうかは、InterSystems IRIS ライセンスによって管理されます。ほとんどの標準 InterSystems IRIS ライセンスが NLP へのアクセスを提供しています。NLP をサポートするどの InterSystems IRIS ライセンスでも、NLP の全コンポーネントを制限なしで完全に使用できます。

NLP は、ObjectScript プログラムから呼び出すことが可能なクラス・オブジェクト・メソッドとプロパティを含む API のコレクションとして提供されます。InterSystems IRIS から NLP 処理を呼び出すための API が提供されています (API クラス)。また、SQL (QAPI クラス) と SOAP Web サービス (WSAPI クラス) から NLP 処理を呼び出すための同等の API が提供されています。これらの API は、“インターシステムズ・クラス・リファレンス” の “%iKnow” パッケージで説明されています。NLP は InterSystems IRIS のコア・テクノロジーであるため、アプリケーションのようなインタフェースは備えていません。ただし、NLP はいくつかの一般的なサンプル出力インタフェースを %iKnow.UI パッケージで提供しています。

InterSystems IRIS NLP を使用してテキスト・ソースを分析するには、InterSystems IRIS ネームスペースに NLP **ドメイン** を定義し、そのドメインにテキストをロードする必要があります。単一のネームスペースに (複数のコンテキストに対する) 複数のドメインを作成できます。ドメインを作成すると、そこにある一連のテキスト・ソースのインデックスが NLP によって生成されます。すべての NLP クエリと他のテキスト処理では、このデータにアクセスするためのドメインを指定する必要があります。

2.1 サンプル・プログラムに関するメモ

このドキュメント内のサンプルの多くで、Aviation.Event SQL テーブルからのデータが使用されています。このサンプル・データを使用する場合、<https://github.com/interSystems/Samples-Aviation> で入手できます (GitHub の知識や GitHub アカウントは必要ありません)。**README.md** ファイルの内容を確認します。このファイルは、GitHub リポジトリに含まれるファイル名とディレクトリの下に表示されています。**README.md** ファイルの下部にある “Setup instructions” セクションまでスクロールして、その手順を完了します。

このドキュメントのサンプル・プログラムの多くでは、最初にドメイン (またはそのデータ) を削除してから、元のテキスト・ファイルから空のドメインにすべてのデータをロードしています。これにより、サンプル・プログラムの目的に沿って、NLP がインデックスを作成したソース・データが、ロード元のファイルや SQL テーブルのコンテンツと正確に一致することが保証されます。

この削除および再ロード手法は、多数のテキスト・ソースを処理する実際のアプリケーションには推奨されません。現実のアプリケーションでは、ドメインの全ソースのロードという時間のかかる作業は、一度だけ実行すべきです。その後で個々のソースを追加または削除して、インデックスが作成されたソース・データを元のテキスト・ファイルや SQL テーブルのコンテンツで最新の状態に保つことができます。

2.2 %Persistent オブジェクト・メソッドに関するメモ

NLP では、ドメインや構成などのオブジェクト・インスタンスの作成と削除を行うために、標準の %Persistent オブジェクト・メソッドをサポートしています。これらの %Persistent メソッドの名前は、%New() などの % 記号で始まります。%Persistent オブジェクト・メソッドの使用は、Create() などの旧式の非永続メソッドを使用する際に推奨されます。新しいコードに対しても、%Persistent オブジェクト・メソッドを使用することをお勧めします。このドキュメント全体を通じたプログラム例は、それら推奨 %Persistent メソッドの使用のために改訂されています。

%New() 永続メソッドでは、%Save() メソッドが必要になることに注意してください。旧式の Create() メソッドでは、個別の保存操作が必要ありません。

2.3 %iKnow と %SYSTEM.iKnow に関するメモ

このドキュメント全体で参照されるすべての NLP クラスは %iKnow パッケージ内にあります。ただし、%SYSTEM.iKnow クラスには、NLP の一般的な処理のコーディングを簡素化するために使用可能な、複数の NLP ユーティリティも含まれています。これらのユーティリティにはショートカットが作成されているため、%SYSTEM.iKnow クラス・メソッドによって実行されるすべてのオペレーションは、%iKnow パッケージ API でも実行できます。

%SYSTEM.iKnow クラス・メソッドの情報を表示するには、Help() メソッドを使用します。すべての %SYSTEM.iKnow メソッドの情報を表示するには、%SYSTEM.iKnow.Help("") を呼び出します。特定の %SYSTEM.iKnow メソッドの情報を表示するには、以下の例に示すように、メソッド名を Help() メソッドに指定します。

ObjectScript

```
DO ##class(%SYSTEM.iKnow).Help("IndexDirectory")
```

詳細は、“インターシステムズ・クラス・リファレンス” を参照してください。

2.4 必要なディスク容量と NLP グローバル

ネームスペースの NLP グローバルは ^IRIS.IK という接頭語を持ちます。

- ・ ^IRIS.IK.* は最終グローバルで、NLP データを含む永続的グローバルです。この NLP データは、元のソース・テキストの約 20 倍のサイズになります。
- ・ ^IRIS.IKS.* はステー징・グローバルです。データのロード時に、これは元のソース・テキストの 16 倍のサイズに増大する可能性があります。ステー징・グローバルは、ジャーナルされていないデータベースにマッピングしてください。ソースのロードと処理が完了すると、NLP はこれらのステー징・グローバルを自動的に削除します。
- ・ ^IRIS.IKT.* は一時グローバルです。データのロード時に、これは元のソース・テキストの 4 倍のサイズに増大する可能性があります。一時グローバルは、ジャーナルされていないデータベースにマッピングしてください。ソースのロードと処理が完了すると、NLP はこれらの一時グローバルを自動的に削除します。
- ・ ^IRIS.IKL.* はログ・グローバルです。これらはオプションで、無視できるほど小さなサイズです。

注意 これらのグローバルは内部使用専用です。いかなる場合も、NLP ユーザは NLP グローバルとの直接の対話を試みてはいけません。

例えば、30GB のソース・ドキュメントをロードする場合は、NLP の永続的なデータ ストレージでは 600GB が必要になります。データのロード時には、1.17TB の使用可能なスペースが必要で、そのうちの 600GB が NLP によるインデックス作成の完了時に自動解放されます。

さらに、Mgr ディレクトリの iristemp サブディレクトリは、ファイルのロード時およびインデックス作成時に元のソース・テキストの 4 倍のサイズに増大する可能性があります。

元のソース・テキストのサイズに基づいて、InterSystems IRIS のグローバル・バッファのサイズを増やす必要があります。詳細は、このドキュメントの“[テキスト・ロード時のパフォーマンスに関する考慮事項](#)”の章を参照してください。

2.4.1 バッチ・ロードのスペース割り当て

InterSystems IRIS では、ソース・テキストの[バッチ・ロード](#)を処理するために、各 NLP ジョブに 256MB の追加メモリが割り当てられます。既定では、NLP によってシステム上の各プロセッサ・コアにジョブが 1 つ割り当てられます。[\\$\\$\\$IKPJOBS ドメイン・パラメータ](#)では NLP ジョブの数を設定し、通常は既定の設定で最適な結果が示されます。ただし、NLP ジョブの最大数は、16 またはプロセッサ・コアの数のうち小さいほうにすることを勧めます。

2.5 入力データ

NLP は構造化されていないデータの分析に使用されます。一般的には、このデータは複数のテキスト・ソース(多くの場合は多数のテキスト)から構成されます。テキスト・ソースは以下のような任意のタイプにすることが可能です。

- ・ 構造化されていないテキスト・データを含むディスク上のファイル (例: txt ファイル)。
- ・ 構造化されていないテキスト・データを含む 1 つ以上のフィールドを持つ SQL 結果セットのレコード。
- ・ 構造化されていないテキスト・データを含む RSS Web フィード。
- ・ 構造化されていないテキスト・データを含む InterSystems IRIS グローバル。

NLP は元のテキスト・ソースの変更や、これらのテキスト・ソースのコピーの作成は行いません。NLP はその代わりに、元のテキスト・ソースの分析結果を正規化およびインデックス付けされた項目として格納し、それぞれの項目に ID を割り当てます。この ID によって、NLP はそのソースを参照できます。ソース、文、パス、CRC、およびエンティティの各レベルで、別個の ID が項目に割り当てられます。

NLP は、Czech/cs (チェコ語)、Dutch/nl (オランダ語)、English/en (英語)、French/fr (フランス語)、German/de (ドイツ語)、Japanese/ja (日本語)、Portuguese/pt (ポルトガル語)、Russian/ru (ロシア語)、Spanish/es (スペイン語)、Swedish/sv (スウェーデン語)、および Ukrainian/uk (ウクライナ語) のテキストをサポートしています。テキストに含まれる言語を指定する必要はありません。また、すべてのテキストや個々のテキストのすべての文が同じ言語である必要もありません。NLP は各テキストの各文の言語を自動的に識別できます。そして、その文に適した言語モデルを適用します。テキストが含む言語および[自動言語識別](#)の実行可否を指定する NLP 構成を定義できます。NLP 構成を使用することで、NLP のパフォーマンスを大幅に向上させることができます。

テキスト・コンテンツの種類 (例: 医師の診断書、新聞記事など) を指定する必要はありません。NLP は、あらゆるコンテンツ・タイプのテキストを自動的に処理します。

2.5.1 ファイル形式

NLP はあらゆる拡張子 (接尾語) を持つあらゆる形式のソース・ファイルを受け入れます。既定では、ソース・テキスト・ファイルはフォーマットされていないテキスト (例: .txt ファイル) から構成されていると想定されます。他の形式のソース・ファイル (.rtf、.doc など) も処理されますが、一部のフォーマット要素がテキストとして処理される可能性があります。これを回避するために、ソース・ファイルを .txt ファイルに変換してこれらの .txt ファイルをロードするか、NLP [コンバータ](#)を作成して NLP ロード時にソース・テキストからフォーマットを除去することができます。

ファイル拡張子のリストを [リスタ・パラメータ](#) として指定します。すると、これらの拡張子を持つファイルのみがロードされます。例えば、このファイル拡張子のリストは、AddListToBatch() メソッド・パラメータとして指定できます。

2.5.2 SQL レコード形式

NLP は SQL 結果セットのレコードをソースとして受け入れます。NLP は NLP ソース ID として各レコードに一意の整数値を生成します。NLP では一意の値を含む SQL フィールドを指定でき、NLP はこれを使用してソース・レコードの外部 ID を作成します。NLP ソース ID と外部 ID は、しばしば両方とも一意の整数値になりますが、NLP ソース ID は NLP によって割り当てられ、外部 ID ではないことに注意してください。一般的に、NLP ソース・テキストは結果セット・レコードのフィールドの一部のみ（多くの場合、構造化されていないテキスト・データを含む 1 つのフィールド）から取得されます。レコードの他のフィールドは無視できます。または、それらの値をメタデータとして使用して [フィルタ処理](#) (包含または除外) したり、ソースにアノテーションを付けることもできます。

2.5.3 テキストの正規化

NLP は元のソース・テキストへのリンクを維持します。これにより、元の大文字の使用や句読点などのままで文を返すことが可能となっています。NLP 内では、マッチングを容易にするために、エンティティに対して以下のような正規化処理が実行されます。

- ・ 大文字と小文字は区別されません。NLP マッチングでは大文字と小文字が区別されません。エンティティ値は、すべて小文字で返されます。
- ・ 余分な空白は無視されます。NLP はすべての単語を 1 つの空白で区切られているものとして扱います。
- ・ 複数のピリオド (...) は 1 つのピリオドに削減され、このピリオドは NLP によって文の終了文字として扱われます。
- ・ ほとんどの句読点は、文、概念、関係を識別するために言語モデルで使用された後で破棄されて、さらなる分析からは除外されます。句読点は一般的にエンティティ内に保持されません。ほとんどの句読点は、句読点記号の前または後に空白がない場合のみ、エンティティに保持されます。ただし、スラッシュ (/) およびアット記号 (@) は、前後の空白の有無に関係なく、エンティティに保持されます。
- ・ 特定の言語特有文字は正規化されます。例えば、ドイツ語の eszett (ß) 文字は “ss” として正規化されます。

NLP エンジンでは、ソース・テキストのインデックスが作成されると、自動的にテキスト正規化を実行します。また、NLP は自動的にディクショナリの項目や用語のテキスト正規化を実行します。

また、いずれの NLP データ・ロードにも依存せずに文字列の NLP テキスト正規化を実行するには、Normalize() または NormalizeWithParams() メソッドを使用できます。詳細は、以下の例を参照してください。

ObjectScript

```
SET mystring="Stately plump Buck Mulligan ascended the StairHead, bearing a shaving bowl"
SET normstring=##class(%iKnow.Configuration).NormalizeWithParams(mystring)
WRITE normstring
```

2.5.4 ユーザ定義のソースの正規化

ユーザはソースの正規化を行うために何種類かのツールを定義できます。

- ・ [コンバータ](#) は、ロード時にソース・テキストを処理してフォーマット・タグや他のテキスト以外のコンテンツを除去します。
- ・ [UserDictionary](#) では、ロード時に特定の入力テキスト・コンテンツ要素を書き換える方法または使用する方法についてユーザが指定できます。例えば、UserDictionary では、既知の省略形や頭文字の置換を指定できます。これは変形や同義語を除去することでテキストを標準化するためによく使用されます。また、NLP による標準の句読点処理に対するテキスト固有の例外を指定するためにも使用できます。

2.6 出力構造

NLP は、その処理の結果を格納するためにグローバル構造を作成します。これらのグローバル構造は、NLP クラス API のみによって使用されることを意図されています。ユーザがこれらを見ることはできず、ユーザがこれらを変更することはできません。

NLP がインデックスを作成したデータは、InterSystems IRIS [リスト構造](#)として格納されます。各 NLP リスト構造には、その項目に対して生成された ID (一意の整数値) が含まれます。NLP エンティティには、値または整数値の ID によってアクセスできます。

NLP はインデックス付けされたエンティティ間の関係を保持して、各エンティティが関連するエンティティ、そのエンティティ・シーケンスのパス、そのパスを含む元の文、およびソース・テキスト内でのその文の位置を参照できるようにしています。元のソース・テキストには、常に NLP からアクセスできます。NLP の処理によって元のソース・テキストが変更されることは一切ありません。

2.7 定数

NLP は %IKPublic.inc ファイルに定数値を定義します。以下の例に示されているように、このインクルード・ファイルを指定した後で、\$\$\$ マクロ呼び出しを使用して、これらの定数を呼び出すことができます。

ObjectScript

```
#include %IKPublic
WRITE "The $$$FILTERONLY constant=", $$$FILTERONLY
```

これらの定数には、ドメイン・パラメータ名、クエリ・パラメータ値、および他の定数が含まれます。

2.8 エラー・コード

一般的なエラー・コードの 8000 ~ 8099 は、NLP での使用に予約されています。詳細は、“InterSystems IRIS エラー・リファレンス”の“[一般的なエラー・メッセージ](#)”を参照してください。

3

ドメイン・アーキテクト

InterSystems IRIS® Data Platform には、NLP ドメインの作成、NLP ドメインへのデータの移入、およびインデックス付きデータに対する分析の実行のために、対話型インタフェースとしてドメイン・アーキテクトがあります。ドメイン・アーキテクトにアクセスするには、InterSystems IRIS 管理ポータルを使用します。

これは、以下の 3 つのツールで構成されています。

- ・ **ドメイン・アーキテクト**：NLP ドメインを作成し、そのドメインにソース・テキスト・データを移入するためのツールです。
- ・ **ドメイン・エクスプローラ**：特定のエンティティに注目することで、NLP ドメイン内のデータを分析するためのツールです。
- ・ **インデックス作成結果**：NLP が、ソースに含まれるテキスト・データをどのように分析したかについて表示するためのツールです。さまざまなタイプのエンティティを示すために強調表示が使用されます。

ドメイン・アーキテクトが提供するすべての機能は、ObjectScript を使用して、NLP クラスのメソッドとプロパティを呼び出すことでも使用できます。

3.1 アーキテクトへのアクセス

ドメイン・アーキテクトにアクセスする場合の開始点は、管理ポータルの **[Analytics]** オプションになります。ここから、**[Text Analytics]** オプションを選択します。これにより、**[ドメイン・アーキテクト]** オプションが表示されます。

すべての NLP **ドメイン**は、**特定のネームスペース内に存在します**。

ネームスペースは、使用する前に、NLP **対応**にしておく必要があります。NLP 対応ネームスペースを選択すると、NLP の **[ドメイン・アーキテクト]** オプションが表示されます。

注釈 NLP 対応ネームスペースを選択しても **[ドメイン・アーキテクト]** オプションが表示されない場合、そのユーザーには有効な NLP ライセンスがありません。管理ポータルのヘッダで、**[ライセンス先]** を調べてください。ライセンス・キーを確認するか、有効にしてください。

3.1.1 ネームスペースの有効化

ネームスペースは、ドメイン・アーキテクトで使用する前に、NLP 対応にしておく必要があります。

- ・ 現在のネームスペースが NLP 対応の場合、**[ドメイン・アーキテクト]** オプションにドメイン・アーキテクトが表示されます。

別の NLP 対応のネームスペースを選択するには、管理ポータル・インタフェース・ページの上にある **[変更]** オプションを使用します。これにより、利用可能なネームスペースのリストが表示され、その中から選択できます。

- ・ 現在のネームスペースが NLP 対応でない場合は、**[Analytics]** オプションに、Analytics 対応のネームスペースのリストが表示されます。表示されたネームスペースの 1 つを選択します。
- ・ NLP 対応のネームスペースが存在しない場合、**[アナリティクス]** オプションには、いずれのオプションも表示されません。

管理ポータルからネームスペースを NLP に対して有効にするには、**[システム管理]**→**[セキュリティ]**→**[アプリケーション]**→**[ウェブ・アプリケーション]** を選択します。これにより、Web アプリケーションのリストが表示されます。3 番目の列には、リストされた項目がネームスペースであるか ([はい])、ネームスペースでないかが示されます。リストから、目的のネームスペース名を選択します。これにより、**[ウェブ・アプリケーション編集]** ページが表示されます。ページの **[有効]** セクションで、**[Analytics]** チェック・ボックスにチェックを付けます。**[保存]** ボタンをクリックします。

%SYS ネームスペースを有効にすることはできません。これは、%SYS ネームスペースにおいて NLP ドメイン作成が不可能となっているためです。

管理ポータルの既定のネームスペースを設定できます。管理ポータルの **[システム管理]**→**[セキュリティ]**→**[ユーザ]** を選択します。目的のユーザの名前を選択します。これにより、ユーザ定義を編集できるようになります。**[一般]** タブで、ドロップダウン・リストから **[開始ネームスペース]** を選択します。**[保存]** をクリックします。

3.2 ドメインの作成

ドメイン・アーキテクトで、**[新規作成]** ボタンをクリックして、ドメインを定義します。以下のドメイン値を記載された順序で指定します。

- ・ **[ドメイン名]** : ドメインに割り当てる名前は、現在のネームスペースに対して一意にする必要があります (パッケージ・クラス内で一意にするだけではありません)。ドメイン名に、長さの制限はありません。また、入力可能な任意の文字 (スペースを含む) を使用できます (% 文字も有効ですが、その使用を避けてください)。ドメイン名は、大文字と小文字が区別されません。ただし、ドメイン・アーキテクトは、既定のドメイン定義クラス名を生成する際に、ドメイン名を使用するため、ドメインに名前を付けるときには、やむを得ない理由があり、ほかに方法がない場合を除いて、クラスの名前付け規約に従うことをお勧めします。
- ・ **[定義クラス名]** : ピリオドで区切られた、ドメイン定義パッケージ名とクラス名。最初にドメイン名を指定した場合は、**[定義クラス名]** をクリックすることで、ドメイン定義パッケージとクラスの既定の名前が生成されます。パッケージ名は、既定で User に設定されます。クラス名は、既定でドメイン名に設定されます (英数字以外の文字は取り除かれます)。この既定値は、そのまま使用することも、変更することもできます。

有効なパッケージ名とクラス名には、英数字のみ使用できます (大文字と小文字が区別されます)。既存のパッケージ名と大文字小文字のみが異なるパッケージ名を指定すると、エラーが発生します。パッケージ内で既存のクラス名と大文字小文字のみが異なるクラス名を指定すると、エラーが発生します。
- ・ **[カスタム更新の許可]** : (オプション) このドメインに、データまたはディクショナリを手動で追加する場合は、これを選択します。既定では、カスタムの更新は許可されません。

[完了] ボタンをクリックして、ドメインを作成します。これにより、**[モデル要素]** の選択画面が表示されます。

新しく作成したドメインは、終了前に **[保存]** して、**[コンパイル]** する必要があります。

重複したドメイン名を作成しようとする、ドメイン・アーキテクトから “このドメイン名はすでに使用されています” というエラーが発行されます。

その他の方法でドメインを作成する場合は、“**NLP ドメイン**” を参照してください。ドメイン・アーキテクトは、ドメイン定義パッケージ名とクラス名を定義できる、単なるドメイン作成インタフェースであることに注意してください。

3.2.1 ドメインを開く方法

管理ポータル・インタフェースを使ってドメインを作成すると、そのドメインが直ちに開き、新しいドメインをすぐに管理できます。

既存のドメインを管理するには、[開く] ボタンをクリックして、ネームスペース内に存在するすべてのドメインをリスト表示します。この表示ではドメインを含むパッケージがリストされます。パッケージを選択してそのドメインを表示します。既存のドメインを選択します。これにより、[モデル要素] の選択画面が表示されます。

3.2.2 ドメイン定義の変更

ドメインを作成したり開いたりすると、[モデル要素] ウィンドウが表示されます。このウィンドウ内でドメイン名をクリックすると、[詳細] タブに、[ドメイン名] フィールド、[ドメイン・テーブル・パッケージ] フィールド、[カスタム更新の許可] および [無効] のチェック・ボックスが表示されます。ドメインに関するこのような特性は、変更が可能です。

- ・ [ドメイン名] を変更すると、モデル要素の表示名が変更されます。[開く] ドメイン・ボタンで表示される [定義クラス名] は変更されません。
- ・ [ドメイン・テーブル・パッケージ] を指定すると、ドメインのデータのテーブル・プロジェクションが指定したパッケージに生成されます。
- ・ [カスタム更新を許可] チェック・ボックスにチェックを付けると、ドメイン・アーキテクト以外のインタフェースを使用して、データ・ソースとディクショナリをこのドメインに手動でロードできるようになります。
- ・ [無効] チェック・ボックスにチェックを付けると、ビルド操作中にいずれのデータ (ソース・データ、メタデータ、ディクショナリ・マッチング・データ) もロードできなくなります。これらのデータ型それぞれに個別の [無効] チェック・ボックスがあり、データ型ごとに個別にロードを無効にできます。

名前を変更したドメインは、終了前に [保存] して、[コンパイル] する必要があります。

3.2.3 ドメインの削除

現在のドメインを削除するには、[削除] ボタンをクリックします。すると、[削除] ドメイン・データ・ウィンドウが表示されます。ドメインの内容だけを削除したり、ドメインの定義を削除したりできます。[ドメインおよび定義クラスの削除] をクリックして、ドメインと、そのドメインに関連付けられたクラス定義を削除します。データ・ソース、skiplist、およびその他のモデル要素の仕様も削除します。

3.3 モデル要素

ドメインの作成後、または既存のドメインを開いた後で、そのドメインのモデル要素を定義できます。モデル要素を追加または変更するには、いずれかの見出しの横にある、展開表示用の三角形をクリックします。初期状態では、展開表示は行われません。いくつかのモデル要素を定義した後で、展開表示用の三角形をクリックすると、定義したモデル要素が表示されます。

モデル要素を追加するには、見出しをクリックします。その後で、右側の [詳細] タブに表示される [追加] ボタンをクリックします。名前と値を指定します。[詳細] の領域から別の場所に移動すると、モデル要素が自動的に生成されます。モデル要素は作成順にリスト表示され、一番最後に作成した要素がリストの先頭に配置されます。モデル要素を変更しても、リスト内での位置は変わりません。

モデル要素を変更するには、見出しを展開して、定義済みのモデル要素をクリックします。現在の値は、右側の [詳細] タブに表示されます。必要に応じて、名前や値を変更します。[詳細] の領域から別の場所に移動すると、モデル要素が自動的に再生成されます。

モデル要素の作成が完了したら、[すべて展開] ボタン（または展開表示用の三角形のいずれか）をクリックして、これらの定義済みの値を表示します。[要素タイプ] 列には、各モデル要素のタイプが表示されます。赤色の“X”をクリックすると、そのモデル要素が削除されます。

[保存] ボタンをクリックすると、すべての変更内容が保存されます。未保存の変更がある場合は、[ドメイン・アーキテクト] のページ見出しの後に、アスタリスク (*) が表示されます。[保存] をクリックして、変更内容を保存します。

[元に戻す] ボタンをクリックすると、直近の未保存の変更が元に戻されます。[元に戻す] を繰り返しクリックして、変更を加えたときと逆の順序で、未保存の変更を元に戻すことができます。一度変更を保存すると、このボタンは表示されなくなります。

以下の [モデル要素] が用意されています。

- ・ [ドメイン設定](#)
- ・ [メタデータ・フィールド](#)
- ・ [データ位置](#)
- ・ [Skiplist](#)
- ・ [マッチング](#)

3.3.1 ドメイン設定

このモデル要素を使用すると、ドメインの特性を変更できます。すべてのドメイン設定はオプションであり、既定値が採用されます。[ドメイン設定] には、以下のオプションがあります。

- ・ **[言語]**：テキスト・データ内で NLP が識別するようにする言語を 1 つ以上選択します。複数の言語にチェックを付けると、[自動言語識別](#) が有効になります。これにより、テキストに対して要求される処理が増加します。そのため、選択した言語のテキストが実際にデータ・セットの一部になる見込みがない場合には、複数の言語を選択しないでください。既定の言語は、英語です。
- ・ **[パラメータの追加]**：このボタンを使用すると、[ドメイン・パラメータ](#) の値を指定できます。ドメイン・パラメータは、空のドメインにのみ追加できます。つまり、目的のドメイン・パラメータをすべて追加してから、[データ位置](#) を指定してドメインを [ビルド](#) する必要があります。そうしないと、ドメイン・パラメータを追加、変更、または削除するための [コンパイル](#) は失敗し、エラー・メッセージが表示されます。[削除](#) ボタンを使用してドメインの内容を削除すると、ドメイン・パラメータを追加、変更、または削除することができます。
 パラメータを追加するには、ドメイン・パラメータの名前と新しい値を指定します。ドメイン・パラメータ名は、大文字と小文字が区別されます。どちらの名前形式も使用できます。例えば、Name=SortField、Value=1 や Name=\$\$\$IKP-SORTFIELD、Value=1 のように指定します。検証は実行されません。すべての未指定のドメイン・パラメータには、既定値が採用されます。追加したパラメータを表示するには、[ドメイン設定] の見出しを展開します。
- ・ **[最大概念長]**：概念としてインデックスを付ける必要のある、単語の最大数です。このオプションは、長いシーケンスの単語が概念としてインデックス付けされないようにするために提供されています。既定値 (0) は、単語の最大数に対する言語固有の既定値を使用します。変更する特別な理由がない限り、この既定値を使用する必要があります。
- ・ **[ユーザ・ディクショナリの管理]**：このボタンを使用すると“[ユーザ・ディクショナリの管理]” ボックスが表示され、そこで 1 つ以上の文字列をユーザ・ディクショナリに指定できます。指定された文字列のそれぞれは、新しい文字列に書き直す文字列を指定するか、ドロップダウン・リストから属性レベルを割り当てる文字列を指定します。

3.3.2 メタデータ・フィールド

[メタデータの追加]：このボタンを使用すると、[ソース・メタデータ・フィールド](#) を指定できます。それぞれのメタデータ・フィールドに、フィールド名、データ型 (String、Number、または Date)、サポートされる操作、およびストレージ・タイプを指定します。ドメインの作成後、必要に応じて、ソースをフィルタ処理するための条件として使用できるメタデータ・フィー

ルドを 1 つまたは複数指定できます。メタデータ・フィールドは、NLP データ・ソースに関連付けられたデータで、NLP がインデックスを作成しないデータです。例えば、テキスト・ソースがロードされた日時は、そのソースのメタデータ・フィールドです。メタデータ・フィールドは、テキスト・データ・ソースをドメインにロードする前に定義する必要があります。

[大小文字区別] チェック・ボックス：既定では、メタデータ・フィールドは大小文字を区別しません。大小文字を区別させるには、このボックスにチェックを付けます。

[無効] チェック・ボックス：[無効] チェック・ボックスにチェックを付けると、すべてのメタデータ・フィールドを無効にできます。または個々のメタデータ・フィールドと共に表示されている[無効] チェック・ボックスにチェックを付けると、そのメタデータ・フィールドのみを無効にできます。無効化したフィールドは、ビルド操作時にロードされません。

ここで指定したメタデータ・フィールドは、[メタデータ・マッピング]というタイトルの下にある、データ位置の詳細項目 **[テーブルからのデータの追加]** および **[クエリからのデータの追加]** に表示されます。

3.3.3 データ位置

追加するデータのソースを指定します。オプションには、**[テーブルからのデータの追加]**、**[クエリからのデータの追加]**、**[ファイルからのデータの追加]**、**[RSS データの追加]**、および **[グローバルからのデータの追加]** があります。

- ・ **[ビルド前に既存のデータを削除する]** チェック・ボックスを使用すると、ここで指定したソース・テキスト・データを追加する前に、この NLP ドメインのインデックス作成済みのソース・テキスト・データを削除するかどうかを指定できます。このチェック・ボックスを使用してデータを削除する場合は、データのロードを無効にすることはできません。新しいデータをロードすることなく、既存のデータを削除するには、**[削除]** ボタンの **[ドメインのコンテンツのみ削除]** オプションを使用します。
- ・ **[無効]** チェック・ボックスを使用すると、ソースのインデックス作成を無効にできます。無効にしたソース・データは、ビルド操作時にロードされません。データのロードが無効にされていると、**[ビルド前に既存のデータを削除する]** チェック・ボックスは無視されます。

多数のテキストに対するビルド操作は、時間がかかる場合があります。すでにデータ位置をロードしてから、メタデータまたはマッチング・ディクショナリを追加または変更したい場合は、データ位置の **[無効]** チェック・ボックスにチェックを付けて、データ位置を再ロードすることなくこれらのモデル要素にインデックスを作成できます。

データ位置を指定した後に、ドメインを **[保存]** および **[コンパイル]** してから、**[ビルド]** ボタンを選択して、データのインデックスをビルドする必要があります。

3.3.3.1 [テーブルからのデータの追加]

このオプションを使用すると、現在のネームスペース内に存在する SQL テーブルに保存されたデータを指定できます。このオプションには、以下のフィールドがあります。

- ・ **[名前]**：抽出対象の結果セット・テーブルの名前を指定するか、そのテーブルの既定の名前を採用します。SQL テーブルの名前付け規約に従ってください。既定の名前は Table_1 です (抽出対象の結果セット・テーブルを追加で定義するたびに、整数がインクリメントされます)。
- ・ **[バッチモード]**：ソース・テキスト・データをバッチ・モードでロードするかどうかを示すチェック・ボックスです。
- ・ **[スキーマ]**：このドロップダウン・リストから、現在のネームスペースに存在するスキーマを選択します。
- ・ **[テーブル名]**：このドロップダウン・リストから、選択したスキーマ内に存在するテーブルを選択します。
- ・ **[ID フィールド]**：このドロップダウン・リストでは、選択したテーブルから ID フィールド (プライマリ・レコード識別子) として機能するフィールドを選択します。ID フィールドには、NULL 以外の一意の値が格納されている必要があります。

ドロップダウン・リストから [] を選択するとフィールド名を入力できます (例えば、**非表示の RowId フィールド** や (まだ) 存在していないフィールド)。フィールド名は、大文字と小文字が区別されません。[] を選択すると、**[既定のオプションを表示する]** ボタンも表示されます。このボタンは、ドロップダウン・リストからテーブルの最初の非表示ではないフィールドを選択します。また、フィールドのドロップダウン・リストに戻ることもできます。

- ・ [グループ・フィールド] : 選択したテーブルからセカンダリ・レコード識別子を取得する SQL select-item 式です。このフィールドは、既定で最初の [ID フィールド] の選択に設定されています。
ドロップダウン・リストから [] を選択するとフィールド名を入力できます (例えば、[非表示の RowId フィールド](#) や (まだ) 存在していないフィールド)。フィールド名は、大文字と小文字が区別されません。[] を選択すると、[\[既定のオプションを表示する\]](#) ボタンも表示されます。このボタンは、ドロップダウン・リストからテーブルの最初の非表示ではないフィールドを選択します。また、フィールドのドロップダウン・リストに戻ることもできます。
- ・ [データ・フィールド] : このドロップダウン・リストでは、選択したテーブルから、データ・フィールドとして機能するフィールドを選択します。データ・フィールドには、NLP インデックス作成に向けてロードしたテキスト・データが格納されます。データ型 %String または %Stream.GlobalCharacter ([文字ストリーム・データ](#)) のフィールドを指定できます。
ドロップダウン・リストから [] を選択するとフィールド名を入力できます (例えば、[非表示の RowId フィールド](#) や (まだ) 存在していないフィールド)。フィールド名は、大文字と小文字が区別されません。[] を選択すると、[\[既定のオプションを表示する\]](#) ボタンも表示されます。このボタンは、ドロップダウン・リストからテーブルの最初の非表示ではないフィールドを選択します。また、フィールドのドロップダウン・リストに戻ることもできます。
- ・ [WHERE 節] : 結果セット・テーブルに取り込むレコードを限定するために、必要に応じて SQL WHERE 節を指定できます。WHERE キーワードは含めないでください。Where 節は検証されません。

このドメインに対して1つ以上の[メタデータ・フィールド](#)を定義済みの場合、[\[メタデータ・マッピング\]](#) オプションでこのテーブルの[メタデータ・フィールド](#)を指定できます。ドロップダウン・リストから、選択されたテーブルのフィールドを選択するか、[\[マップなし\]](#)を選択するか、[\[カスタム\]](#)を選択できます。[\[カスタム\]](#)を選択すると、アーキテクトは空のフィールドを表示します。このフィールドに、カスタム・マッピングを指定できます。

このドメインに対して[メタデータ・フィールド](#)が定義されていない場合、[\[メタデータ・マッピング\]](#) オプションには、[\[メタデータの追加\]](#) ドメイン・オプションを表示する [\[メタデータの宣言\]](#) ボタンが用意されています。

3.3.3.2 [クエリからのデータの追加]

[\[クエリからのデータの追加\]](#) は、[\[テーブルからのデータの追加\]](#) とほとんど同じですが、既存のテーブル (複数可) に対する整形形式の SQL クエリを指定できます。このテーブルから、以下のフィールドを指定します。

- ・ [名前] : 抽出対象の結果セット・テーブルの名前を指定するか、そのテーブルの既定の名前を採用します。SQL テーブルの名前付け規約に従ってください。既定の名前は Query_1 です (抽出対象の結果セット・テーブルを追加で定義するたびに、整数がインクリメントされます)。
- ・ [バッチモード] : ソース・テキスト・データをバッチ・モードでロードするかどうかを示すチェック・ボックスです。
- ・ [SQL] : クエリ・テキストです (InterSystems SQL SELECT 文)。クエリを定義すると、JOIN 構文を使用して、複数のテーブルからフィールドを選択できます。複数のテーブルを指定する場合は、選択したフィールドに列エイリアスを割り当てます。また、クエリを定義すると、グループ・フィールドとして使用可能な式フィールドを指定できます。

クエリ文字列に構文エラーが含まれていたり、存在しない項目を参照している場合、[\[保存\]](#) ボタンを押すと SQLCODE エラーとメッセージが表示されます。エラーを確認すると、クエリが保存されます。SQL クエリ文字列フィールドに戻って、エラーを修正します。

以下のフィールド選択ドロップダウン・リストには、選択したフィールドが表示されます。テーブル・エイリアス接頭語は表示されません。フィールドに列エイリアスがある場合、フィールド名ではなく、このエイリアスがリストされます。

- ・ [ID フィールド] : このドロップダウン・リストでは、選択したテーブルから、ID フィールドとして機能するフィールドを選択します。ID フィールドには、NULL 以外の一意の値が格納されている必要があります。
- ・ [グループ・フィールド] : このドロップダウン・リストでは、クエリからセカンダリ・レコード識別子 (グループ・フィールド) として機能する select-item 式 (SQL 関数式など) を選択します。例えば、YEAR(EventDate) などです。
- ・ [データ・フィールド] : このドロップダウン・リストでは、選択したテーブルから、データ・フィールドとして機能するフィールドを選択します。データ・フィールドには、NLP インデックス作成に向けてロードしたテキスト・データが格納されます。

このドメインに対して 1 つ以上の**メタデータ・フィールド**を定義済みの場合、**[メタデータ・マッピング]** オプションで、各定義済みの**メタデータ・フィールド**に対して**[マップなし]** または **[カスタム]** のどちらかを選択できます。既定値は**[マップなし]** です。**[カスタム]** を選択すると、アーキテクトは空のフィールドを表示します。このフィールドに、**カスタム・マッピング** を指定できます。

このドメインに対して**メタデータ・フィールド**が定義されていない場合、**[メタデータ・マッピング]** オプションでは、**[メタデータの追加]** ドメイン・オプションを表示する**[メタデータの宣言]** ボタンが用意されています。

[モデル要素] ウィンドウの [要素タイプ] 列には、FROM 節内の最初のテーブル名以降が切り詰められた形式で、定義したクエリが表示されます。完全なクエリは、[詳細] ウィンドウに表示されます。

3.3.3.3 ファイルからデータを追加

このオプションを使用すると、ファイルに保存されたデータを指定できます。このオプションには、以下のフィールドがあります。

- ・ **[名前]**：抽出対象のデータ・ファイルの名前を指定するか、そのファイルの既定の名前を採用します。既定の名前は File.1 です (抽出対象のデータ・ファイルを追加で定義するたびに、整数がインクリメントされます)。
- ・ **[パス]**：目的のファイルが格納されているディレクトリへの、完全なディレクトリ・パスです。パスの構文は、ファイルシステムによって異なります。Windows システムの場合は、以下ようになります。C:\temp\NLPSources\
- ・ **[拡張子]**：ファイルの拡張子です (txt や xml など)。ファイル拡張子を指定する場合は、先頭のドットを含めないでください。複数の拡張子は、コンマ区切りリストとして指定します。例えば、txt,xml のようにドットやスペースは含めません。これを指定すると、指定した拡張子を含むファイルのみが、結果として抽出されるデータに含まれます。**[拡張子]** フィールドを空白 (既定) のままにすると、拡張子に関係なく、すべてのファイルが含まれます。
- ・ **[エンコーディング]**：ファイルの処理に使用する、文字セット・エンコーディングの種類のドロップダウン・リストです。
- ・ **[フィルタ条件]**：結果として抽出されるデータに含めるファイルを制限するために使用する条件です。
- ・ **[再帰]**：再帰的にファイルを選択するかどうかを示すチェック・ボックスです。チェックを付けると、指定したディレクトリに含まれるファイルと、そのディレクトリのすべてのサブディレクトリに含まれるファイルからデータを抽出できます。チェックを付けない場合は、指定したディレクトリ内のファイルからのみデータを抽出できます。既定は、非再帰です (チェック・ボックスにチェックが付いていません)。
- ・ **[バッチモード]**：ソース・テキスト・データをバッチ・モードでロードするかどうかを示すチェック・ボックスです。

3.3.3.4 [RSS データの追加]

このオプションを使用すると、RSS ストリーム・フィードからのデータを指定できます。このオプションには、以下のフィールドがあります。

- ・ **[名前]**：抽出対象のデータの名前を指定するか、そのデータの既定の名前を採用します。既定の名前は RSS_1 です (抽出対象の RSS ソースを追加で定義するたびに、整数がインクリメントされます)。
- ・ **[バッチモード]**：ソース・テキスト・データをバッチ・モードでロードするかどうかを示すチェック・ボックスです。
- ・ **[サーバ名]**：URL が検出されたホスト・サーバの名前です。
- ・ **[URL]**：実際の RSS フィードへのサーバ・アドレスに含まれるナビゲーション・パスです。
- ・ **[テキスト要素]**：RSS フィードからロードするテキスト要素のコンマ区切りリストです。例えば、title,description などです。既定値は、空です。

3.3.3.5 [グローバルからのデータの追加]

このオプションを使用すると、InterSystems IRIS グローバルからのデータを指定できます。このオプションには、以下のフィールドがあります。

- ・ [名前]：抽出対象のデータの名前を指定するか、そのデータの既定の名前を採用します。既定の名前は Global_1 です (抽出対象のグローバル・ソースを追加で定義するたびに、整数がインクリメントされます)。
- ・ [バッチモード]：ソース・テキスト・データをバッチ・モードでロードするかどうかを示すチェック・ボックスです。
- ・ [グローバル参照]：ソース・データを抽出するグローバルです。
- ・ [開始添え字]：取り込む添え字の範囲で最初のグローバル添え字です。
- ・ [終了添え字]：取り込む添え字の範囲で最後のグローバル添え字です。
- ・ [フィルタ条件]：結果として抽出されるデータに含めるファイルを制限するために使用する条件です。

3.3.4 Skiplist

skiplist の定義：ドメインの作成後、必要に応じて、そのドメインに 1 つまたは複数の **skiplist** を作成できます。**skiplist** は、クエリから返されることを望まない用語 (単語や語句) のリストです。したがって、**skiplist** を使用して、ドメインにロードされたデータ・ソース内の特定の用語を無視する NLP 処理を実行できます。

- ・ [名前]：新しい **skiplist** の名前を指定するか、既定の名前を採用します。**skiplist** 名では大文字と小文字が区別されません。重複する **skiplist** 名を指定すると、コンパイル・エラーが発生します。既定の名前は Skiplist_1 です (追加の **skiplist** を定義するたびに、整数がインクリメントされます)。
- ・ [エントリ]：**skiplist** に含める用語を指定します (行ごとに 1 つの用語)。用語は、小文字にする必要があります。重複する用語は許容されます。**skiplist** 間で用語のコピーおよび貼り付けを実行できます。用語のグループを区切るために、空白行を含めることができます。用語リストの末尾に改行を入れるかどうかは任意です。空白行はエントリと見なされません。

skiplist を追加、変更、または削除する場合、その変更を有効にするには、ドメインを [保存] および [コンパイル] する必要があります。

skiplist の定義が、ドメインへのデータのロード方法に影響を与えることはないため、**skiplist** に対する変更ではドメインの再構築は必要ありません。

skiplist は、コンパイルされてから、[ドメイン・エクスプローラ](#)に渡されます。ドメイン・エクスプローラでは、ドメインにロードしたソース・テキスト・データの分析時に、1 つまたは複数の **skiplist** を指定できます (指定しなくてもかまいません)。**skiplist** は、一部の (すべてではない) ドメイン・エクスプローラの分析に適用されます。

3.3.5 マッチング

[マッチング] オプションには、ディクショナリを定義し、その項目と用語を指定する [\[ディクショナリの追加\]](#) オプションが用意されています。

[マッチング] オプションには、以下の 4 つのチェック・ボックス・オプションが用意されています。

- ・ [無効]：[無効] チェック・ボックスにチェックを付けると、すべてのディクショナリを無効にできます。または個々のディクショナリと共に表示されている [無効] チェック・ボックスにチェックを付けると、そのディクショナリのビルドを無効にできます。個別の [無効] チェック・ボックスにチェックを付けることによって、変更したディクショナリのみをビルドできます。既定はオフです。
- ・ [構築前に削除]：既定はオン
- ・ [自動実行]：既定はオン
- ・ [辞書エラーを無視]：既定はオン

3.3.5.1 デクショナリの追加

[デクショナリの追加] ボタンは、デクショナリ名 (既定値を表示)、オプションの説明、NLP サポート言語のドロップダウン・リストから選択されたデクショナリ言語、および無効のチェック・ボックスというデクショナリ定義オプションを表示します。既定の名前は Dictionary_1 です (追加のデクショナリを定義するたびに、整数がインクリメントされます)。

[項目の追加] ボタンは、項目名 (既定値を表示)、URI 名 (既定値を表示)、NLP サポート言語のドロップダウン・リストから選択された項目言語、および無効のチェック・ボックスという項目定義オプションを表示します。さらに項目を定義するには、デクショナリ名を選択します。項目は作成順にリストされます (一番最後に作成されたものがリストの先頭にきます)。各項目内で 1 つ以上の用語を定義できます。既定の名前は Item_1、既定の URI 名は uri:1 です (このデクショナリ用に追加の項目を定義するたびに、整数がインクリメントされます)。

[用語の追加] ボタンは、用語を指定する文字列、NLP サポート言語のドロップダウン・リストから選択された用語言語、および無効のチェック・ボックスという用語定義オプションを表示します。さらに用語を定義するには、項目名を選択します。用語は作成順にリストされます (一番最後に作成されたものがリストの先頭にきます)。

3.4 保存、コンパイル、およびビルド

ドメインは、用意されているボタンを使用して、保存、コンパイル、およびビルドする必要があります。モデル要素を追加、変更、または削除した後には、ドメインを保存してコンパイルする必要があります。

[保存] ボタンでは、現在のドメイン定義を保存します。ドメイン定義が開かれていない場合は、アーキテクトにより [保存] ボタンが灰色表示 (無効化) されます。アーキテクトは、変更していないドメイン定義を保存しても、エラーを発行しません。

[コンパイル] ボタンでは、現在のドメイン定義をコンパイルします。ドメイン定義を構成するクラスとルーチンのすべてがコンパイルされます。ドメイン定義に加えた変更を保存していない場合は、コンパイル処理により、コンパイル前にドメイン定義を保存するように促すプロンプトが表示されます。

[ビルド] ボタンでは、指定したソースを現在のドメインにロードします。データ位置、メタデータ・フィールド、またはマッチング・デクショナリに変更を加えた場合は、ドメインをビルドする必要があります。[ドメインのビルド] ウィンドウには、以下に示すような進行状況メッセージが表示されます。

```
13:50:48: Loading data...
13:51:49: Finished loading 3 sources
13:51:49: Creating dictionaries and profiles...
13:51:49: Finished creating 1 dictionaries, 1 items, 3 terms and 0 formats
13:51:49: Matching sources...
13:51:50: Finished matching sources
13:51:50: Successfully built domain 'mydomain'
```

ビルド処理には時間がかかる場合があります。特定のモデル要素の [無効] チェック・ボックスにチェックが付けられていると、それに対応するソースは、ビルド処理でロードされません。個別の [無効] チェック・ボックスにチェックを付けることによって、変更したモデル要素のみをビルドできます。

3.5 ドメイン・エクスプローラ

ドメイン・エクスプローラには、以下の 2 つの方法でアクセスできます。

- 管理ポータル の [Analytics] オプションで、[Text Analytics] オプションを選択します。これにより、[ドメイン・エクスプローラ] オプションが表示されます。このオプションを選択すると、ドロップダウン・リストから既存のドメインを選択するように求められます。

- ・ 管理ポータルの **[Analytics]** オプションで、**[Text Analytics]** オプションを選択します。ドメイン・アーキテクトにアクセスして、ドメインを作成するか、ドメインにアクセスします。データ位置を指定し、**[ビルド]** ボタンを使用して、対象のデータをドメインに入力すると、**[ツール]** タブから **[ドメイン・エクスプローラ]** を選択できます。これにより、現在のドメインが選択された状態で、ドメイン・エクスプローラが個別のブラウザ・タブとして表示されます。

ドメイン・エクスプローラは、幅広い用途を持つ表示インタフェースです。ドメイン内でインデックス作成済みのソース・テキスト・データについて、豊富な情報が表示されます。これは、最初に上位の（最も頻繁に出現する）概念、または優位な（最も優位性の高い）概念のいずれかのリストを表示します。この 2 つのリストを切り替えることができます。

エンティティを選択すると、**[ドメイン・エクスプローラ]** は、類似エンティティと関連概念の分析、およびより大きなテキスト・ユニット（ソース、パス、および CRC）における指定したエンティティの出現についての分析を提供します。ここでは、データに含まれる内容について一目でわかるコンテキスト・ビューが示されます。

[ドメイン・エクスプローラ] には、メタデータ条件に基づいて、ドメインに含まれるソースのサブセットの選択をサポートする、汎用フィルタがあります。このインタフェースは、NLP スマート・インデックス作成を使用して大量のドキュメントの概観と移動をすばやく行う方法のサンプルを示しています。

3.5.1 ドメイン・エクスプローラの設定

既定では、**[ドメイン・エクスプローラ]** には、**[ドメイン・エクスプローラ]** を呼び出したときにドメイン・アーキテクトで現行だったドメインまたは選択していたドメインの分析が表示されます。

別のドメインを選択する手順は以下のとおりです。

1. **[ドメイン・エクスプローラ]** の右上にあるギアのアイコンを選択します。**[設定]** ボックスが表示されます。
2. **[設定]** ボックスには、**[ドメインの切り替え]** ドロップダウン・リストが含まれています。このリストからドメインを選択します。既定では、このリストには、現在のネームスペースで定義されているドメインが表示されます。**[他のネームスペースを含める]** チェック・ボックスにチェックを付けると、すべてのネームスペースで定義されているドメインがドロップダウン・リストに表示されます。

skiplist を適用する手順は以下のとおりです。

1. **[ドメイン・エクスプローラ]** の右上にあるサングラスのアイコンを選択します。定義されている skiplist がドメインにならない場合、このアイコンは表示されません。
2. **[Skiplists]** ボックスには、定義されているそれぞれの skiplist についてチェック・ボックスが用意されています。1 つ以上にチェックを付け、**[適用]** ボタンをクリックします。

語幹解析を使用する手順は以下のとおりです。

1. **[ドメイン・エクスプローラ]** の右上にあるギアのアイコンを選択します。**[設定]** ボックスが表示されます。
2. ドメインに語幹解析が構成されている場合、**[設定]** ボックスには、**[エンティティの代わりに語幹を使用する]** チェック・ボックスと **[語幹の表現形式を表示する]** チェック・ボックスも含まれています。**[エンティティの代わりに語幹を使用する]** にチェックを付けると、**[ドメイン・エクスプローラ]** によって語幹解析が実行され、**[ドメイン・エクスプローラ]** の見出しが以下のように変更されます。**[上位の概念]/[優位な概念]** は **[上位の語幹]/[優位な語幹]** になり、**[類似エンティティ]** は **[類似語幹]** になり、**[関連概念]** は表示されなくなり、**[近似プロファイル]** はそのまま、**[CRC]** タブは表示されなくなります。**[語幹の表現形式を表示する]** にチェックを付けると、それぞれの語幹が代表語として表示されます。チェックを外すと、語幹自体が表示されます。既定では、両方のチェック・ボックスにチェックが付いています。

[ドメイン・エクスプローラ] の右上の数字は、選択したドメインでロードされたソースの数で、データ分析に使用できます。**フィルタ** を適用することでこの数字を制限できます。

3.5.2 すべての概念のリスト

ドメイン・エクスプローラは最初、ドメインにロードされたデータ・ソースについて概念の分析を提供します。概念をリストする方法は、頻度別と優位性別の 2 つあります。この 2 つを切り替えることができます。そのためには、[頻度] または [優位性] のいずれかのボタンを選択します。

- ・ **[上位の概念]** : [頻度] ボタンを選択すると、ソースのすべての概念が**頻度**の降順でリスト表示されます。複数の概念が同じ頻度の場合、文字列照合の降順でリスト表示されます。各概念は、頻度 (すべてのソース内での合計出現回数) および分布 (その概念を含んでいるソースの数) と共にリストされます。1 つのソースに対する頻度のカウント数を表示するには、**インデックス作成結果**ツールを使用します。
- ・ **[優位な概念]** : [優位性] ボタンを選択すると、ソースのすべての概念が**優位性スコア**の降順でリスト表示されます。複数の概念が同じ優位性スコアの場合、文字列照合の降順でリスト表示されます。優位性スコアは、ソースごとの優位性の値を考慮に入れ、ロードされたすべてのソースにわたる概念の優位性を判断する平均化アルゴリズムを使用して計算されます。1 つのソースにおける優位性の値は、整数値になります。最も優位性の高い概念には、優位性に 1000 が割り当てられます。1 つのソースに対する優位性の値を表示するには、**インデックス作成結果**ツールを使用します。

3.5.3 指定したエンティティの分析

特定のエンティティの分析を表示する方法は 2 つあります。

- ・ **[上位の概念]** または **[優位な概念]** のリストのいずれかから概念を選択します。
- ・ 左上隅にある入力フィールドでは、エンティティに含まれる単語の先頭の数文字 (最小 2 文字、大文字小文字の区別なし) を入力でき、ドメイン・エクスプローラには、その文字で始まる単語を含むすべての既存のエンティティのドロップダウン・リストが表示されます。このドロップダウン・リストからエンティティを選択して、[エクスプローラ] ボタンをクリックします。このオプションを使用して関係または概念を表示できます。両方のタイプのエンティティがドロップダウン・リストに表示されます。

1 つのエンティティを選択すると、そのエンティティについて**関連するエンティティ**と**コンテキストで指定されるエンティティ**の 2 種類の分析が表示されます。

3.5.3.1 関連するエンティティ

1 つのエンティティを選択すると、以下のリストが表示されます。

- ・ **[類似エンティティ]** : 指定したエンティティに**似ている**概念と関係、およびそれぞれの概念または関係の頻度 (すべてのソース内での合計出現回数) および分布 (その概念を含んでいるソースの数) のリスト。先頭にリストされる類似エンティティは、常に指定したエンティティ自体になります。概念については、先頭にリストされるこのエンティティが、その概念の**[上位の概念]** リストと同じになります。
- ・ **[関連概念]** : [関連] ボタンを選択すると、指定した概念に**関連する**概念のリストがそれぞれの概念の頻度 (すべてのソース内での合計出現回数) および分布 (その概念を含んでいるソースの数) と共に表示されます。関連概念とは、指定した概念で CRC に出現する概念のことです。
- ・ **[近似プロファイル]** : [近似] ボタンを選択すると、近似プロファイル・テーブルが表示されます。このテーブルには、指定した概念との**近似**によって関連付けられた概念がリストされます。それぞれの概念の近似スコアも示されます。

[類似エンティティ]、[関連概念]、または [近似プロファイル] のリストからエンティティを選択すると、すべてのリストがそのエンティティの分析に変更されます。**[上位の概念]** と **[優位な概念]** のリストは変更されません。

3.5.3.2 コンテキストのエンティティ

エンティティを選択すると、コンテキストのそのエンティティについて以下のリストも表示されます。

- ・ **[ソース]**：指定したエンティティ（緑色で強調表示）を含むソース・テキストのリスト、内部のソース ID（整数）および **外部のソース ID**。ソースは、内部のソース ID 順（降順）でリストされます。ソース・テキストは、ソースにありこのエンティティを含むすべての文を表示します。間にあるこのエンティティを含まない文は表示されませんが、省略記号 (...) で示されます。最初に表示される文がソースの最初の文ではない場合には先頭の省略記号は表示されませんが、最後に表示される文が実際にソースの最後の文の場合でも、最後の文の後に末尾の省略記号が常に表示されます。

赤いテキストは**否定**を示し、否定属性の範囲内ではエンティティは赤い文字になります。否定の範囲は、対応するパス、文、または CRC と同じであるとは限りません。

[目のアイコン] を選択またはソースのリストのどこかをクリックすると、**ソースのフルテキスト**が表示されます。フルテキストでは、指定したエンティティは出現するたびに強調表示され、否定範囲のそれぞれのテキストは赤い文字で表示されます。（指定したエンティティのすべての出現をこのフルテキスト・ボックスで表示するには、% オプションを 100% に設定する必要があります。）

[矢印アイコン] を選択すると、**インデックス作成結果**ツールが表示されます。

- ・ **[パス]**：指定したエンティティを含むパスのリスト。パスは、ID 順（降順）でリストされます。パス ID はソース単位で割り当てられるため、同じパス・テキストが、複数回、異なるパス ID でリストされることがあります。

パスのエンティティと属性は、“**[インデックス付きの文]**” のインデックス作成結果ツールの説明に記載されているように色分けして強調表示されます。また、**[エクスプローラ]** エンティティは黄橙で表示されます。

パス要素を選択すると、すべてのリストがそのエンティティの分析に変更されます。**[上位の概念]**と**[優位な概念]**のリストは変更されません。

[目のアイコン] を選択すると、**ソースのフルテキスト**が指定エンティティと共に緑色に強調表示されて表示されます。

[矢印アイコン] を選択すると、**インデックス作成結果**ツールが表示されます。

- ・ **[CRC]**：指定したエンティティを含む CRC（概念 - 関係 - 概念）シーケンス、頻度（すべてのソース内でのその CRC の合計出現回数）および分布（その CRC を含んでいるソースの数）のリスト。多くの CRC に含まれる概念は 1 つのみです（CR または RC）。エンティティ・タイプの強調表示は、**[パス]** の強調表示と同じですが、**パス関係の単語**は CRC の一部ではないため、表示されません。属性は、**[CRC]** のリストでは強調表示されません。

CRC 要素を選択すると、すべてのリストがそのエンティティの分析に変更されます。**[上位の概念]**と**[優位な概念]**のリストは変更されません。

[目のアイコン] を選択すると、**[選択した CRC を持つソース]** ボックスが表示され、その CRC のインスタンスを含むそれぞれのソースがリスト表示されます。CRC は、その文のコンテキストで緑色に強調表示され、そのソースのソース ID のフラグが立てられます。ソース ID のリストには、指定した CRC を含む複数の文を含めることができます。間にあるこの CRC を含まない文は省略記号で示されます。**[選択した CRC を持つソース]** ボックスから**[目のアイコン]**を選択して、CRC を含むソースで**ソースのフルテキスト**を指定エンティティ（CRC ではない）と共に緑色に強調表示して表示できます。

注釈 ドメインでサポートされる唯一の言語が日本語の場合は、ドメイン・エクスプローラの表示が異なり、**[関連概念]** および **[CRC]** のリストは表示されません。**[エンティティ・ベクトル]** のリストが、**[パス]** のリストの代わりになります。

3.5.3.3 フルテキスト・ボックス

[目のアイコン] では、選択したソースのフルテキストが表示されます。このテキスト・ボックスはソースの**外部 ID** で識別されます。例えば :SQL:1171:1171 です。

ソースのテキストは、以下のようにタグ付けされます。

- ・ 指定したエンティティは緑色で強調表示されます。
- ・ 赤いテキストは**否定**を示し、否定属性の範囲内ではエンティティは赤い文字になります。

このフルテキスト・ボックスには以下のオプション・ボタンが用意されています。

- ・ **[メタデータ]**：ソースのメタデータを表示します。すべてのソースに **DateIndexed** メタデータ・フィールドがあります。この日付スタンプは、現在のロケールの表示形式で UTC 日付/時刻として表されます。小数部は切り捨てられます。ソース・テキストに戻るには、再度 **[メタデータ]** ボタンをクリックします。
- ・ **[強調表示]**：何も実行しません。
- ・ **[インデックス作成]**：ソースのテキストを強調表示し、以下のようにエンティティのタイプを示します。
 - 緑：指定したエンティティ (概念または関係)。
 - 青：概念。
 - 白：関係。
 - 薄い青：**パス関係の単語**。
 - イタリック：無関係な単語。

否定範囲のテキストは赤い文字で表示されます。

- ・ **[ディクショナリ]**：何も実行しません。
- ・ **[%]**：ソースのテキストを要約します。既定のパーセントは 100% (フルテキスト) です。100 未満の整数を指定してから **[%]** ボタンをクリックすると、ソースの他の文と比べて関連性スコアが低い文を削除することによって、指定したサイズ近くにテキストが減らされ、**ソースのテキストが要約**されます。要約によって、指定したエンティティを含む文が保持されるとは限りません。

3.5.4 分析するソースの制限

データ分析の範囲は、フィルタを使用して制限できます。フィルタを使用すると、ドメインにロードされたデータ・ソースが分析に包含または除外されます。既定では、**[ドメイン・エクスプローラ]** は、ドメインにロードされたすべてのデータ・ソースを分析します。

- ・ ドメイン・エクスプローラの右上にある **[フィルタ・アイコン]** (漏斗) ボタンをクリックするとフィルタが適用され、指定した条件に基づいて分析からソースが包含または除外されます。いくつかの種類のフィルタを指定できます。また、複数のフィルタを適用できます。複数のフィルタを AND、OR、NOT AND、または NOT OR の各ロジックを使用して関連付けることができます。

フィルタを追加するには、ドロップダウン・リストからフィルタのタイプを選択し、フィルタの条件を指定してから **[追加]** ボタン、**[適用]** ボタンの順に選択します。複数のフィルタを追加する場合、**[追加]** ボタンを選択してから AND/OR ロジック・オプションを選択してフィルタを関連付け、**[適用]** ボタンを選択します。

1 つ以上のフィルタが有効になると、**[フィルタ・アイコン]** が緑色で表示されます。

[フィルタ・アイコン] の左側の数字は、フィルタの適用後に取り込んだソースの数を示しています。フィルタが適用されていない場合、この数字はドメイン内のソースの合計数です。

- ・ 1 つのフィルタを削除するには、**[フィルタ・アイコン]** を選択し、フィルタの説明の横にある黒い **[X]** を選択してから **[適用]** ボタンを選択します。すべてフィルタを削除するには、**[フィルタ・アイコン]** を選択し、**[クリア]** ボタンを選択してから **[適用]** ボタンを選択します。

以下のタイプのフィルタがサポートされています。

- **[メタデータ]**：メタデータの値で、ソースを除外するために使用します。既定では、すべてのソースに **DateIndexed** **メタデータ**があります。DateIndexed メタデータを適用するには、このフィールドを選択し、演算子を選択して、カレンダー・アイコンをクリックして日付値を選択してから、目的の日を選択します。
- **[ソース ID]**：取り込むソースを**ソース ID** で選択する場合に使用します。単一のソース ID、またはソース ID のコンマ区切りのリストを指定できます。

- [ソース ID の範囲]：取り込むソースをソース ID で選択する場合に使用します。値の範囲を指定することによって、ソース ID の範囲を指定できます。指定値も範囲に含まれます。
- [外部 ID]：取り込むソースを外部 ID で選択する場合に使用します。例えば :SQL:1171:1171 です。単一の ID、またはコンマ区切りの ID のリストを指定できます。外部ソース ID は[ソース]リストにリスト表示されます。
- [SQL]：SQL クエリを指定して取り込むソースを選択する場合に使用します。

3.6 インデックス作成結果

[インデックス作成結果] ツールを使用すると、個々のデータ・ソースの内容の NLP インデックスを表示できます。これにより、[インデックス付きの文]、[概念]、および [CRC] という 3 つのリストが表示されます。[インデックス付きの文] の表示には、エンティティ・タイプ (概念、関係、無関係、パス関係) を示す色分けされたテキストと、属性およびその範囲を示す色分けされた強調表示の両方が含まれます。

[インデックス作成結果] ツールにアクセスするには、InterSystems IRIS 管理ポータルから [Analytics]、[Text Analytics] の順に選択します。[Analytics] のオプションが表示されるのは、[Analytics] が有効になっているネームスペース内にいる場合のみです。目的のネームスペースを選択します。これにより、[Analytics] のツールが表示されます。以下に示す 2 つの方法のいずれかで [インデックス作成結果] ツールにアクセスできます。

- ・ [Text Analytics]、[インデックス作成結果] オプションの順に選択します。
- ・ [Text Analytics]、[ドメイン・アーキテクト] オプションの順に選択します。ドメイン・アーキテクトで既存のドメインを開くか、新しいドメインを定義します。コンパイル済みのドメインでは、[ツール] タブの [インデックス作成結果] ボタンを使用して、NLP でデータのインデックスがどのように作成されたかを表示できます。この場合、[インデックス作成結果] ツールが個別のブラウザ・タブとして表示されます。

[インデックス作成結果] ツールを使用すると、指定したドメイン内のデータまたは手動入力データのインデックス作成結果を表示できます。

注釈 右上にある [インデックス作成結果] のオプションを表示するには、水平方向にスクロールしなければならない場合があります。

3.6.1 ドメイン・データ

[インデックス作成結果] ウィンドウの右上には、定義されているドメインのドロップダウン・リストがあります。既定では、最初に定義されたドメインが表示されます。目的のドメインを選択します。

ウィンドウの上部にある幅の広い空白のボックスをクリックすると、インデックスが作成された各データ・ソースの内容の単一行ドロップダウン・リストが表示されます。これらのソースのいずれかを選択すると、そのソースのインデックス作成結果が表示されます。

[>>] ボタンを使用すると、幅の広い単一行ソース・ボックスを折りたたむ (非表示にする) ことができます。これにより、水平方向にスクロールせずにインデックス作成結果を表示できるようになります。[<<] ボタンを使用すると、幅の広い単一行ソース・ボックスを空白のボックスとして展開 (再表示) することができます。その空白のボックスをクリックして、別のデータ・ソースを選択できます。

3.6.2 手動入力データ

NLP インデックス作成結果分析用のテキストを直接入力するには、[インデックス作成結果] ウィンドウの右上にある [手動入力] ボタンを選択します。これにより、[リアルタイム入力] ボックスが開きます。空白のボックスに入力テキストを入力

するか、貼り付けます。[構成]ドロップダウン・ボックスを使用して既存（または既定）の構成を選択するか、[言語 →] を選択した後、2 つ目のドロップダウン・リストを使用して各国言語または[自動検出]を選択します。

3.6.3 [インデックス付きの文]

ソース内の文は、1 行に 1 文ずつ順にリストされます。エンティティ・タイプ（概念、関係、無関係、パス関係）と属性は、色分けと強調表示によって示されます。

[インデックス作成結果] ウィンドウの右上で、[強調表示] タイプ ([ライト] または [フル]) を選択できます。[ライト] では、色分けと下線を使用してエンティティ・タイプと属性が示されます。これは、強調を抑えて文を読みやすくすることを目的としています。[フル] では、NLP のインデックス付き構造をより明確に示すために、各エンティティの周りにボックスが表示され、属性には太線が使用されます。どちらのタイプの強調表示でも情報の内容は同じです。既定値は [フル] です。

文のテキストは、エンティティについては以下のように強調表示されます。

- ・ 概念：青、ボックス付き
- ・ 関係：薄緑、ボックス付き
- ・ 無関係：グレー、ボックスなし
- ・ パス関係：黒、グレーのボックス

文のテキストは、属性については以下のように強調表示されます。

- ・ 否定属性の語句には赤いテキストが使用されます（概念は太字、関係は通常の文字で示されます）。概念と関係は、[フル] の強調表示でさらに明確化されます。周りのボックスは、エンティティ・タイプの色です（概念の場合は青、関係の場合は薄緑）。否定のキーワードには、赤い下線が引かれます。複数の単語による否定の用語（“was not” など）は、各単語に赤い下線が引かれた状態で表示されます。
- ・ 時間、期間、または頻度属性の語句には、オレンジの点線で下線が引かれます。時間属性のキーワードには、オレンジの下線が引かれます。期間属性のキーワードには、明るい緑の下線が引かれます。頻度属性のキーワードには、黄色の下線が引かれます。
- ・ 測定属性には、マゼンタの点線で下線が引かれます。測定のキーワードには、マゼンタで下線が引かれます。
- ・ 否定的な感情属性には、紫の点線で下線が引かれます。この感情のキーワードには、紫で下線が引かれます。
- ・ 肯定的な感情属性には、緑の点線で下線が引かれます。この感情のキーワードには、緑で下線が引かれます。

これらを組み合わせることで、エンティティと属性の組み合わせを強調表示することができます。例えば、否定属性の語句に含まれる測定属性などです。

3.6.4 [概念] および [CRC]

[インデックス作成結果] には、ソース内のすべての概念を示すリストと、ソース内のすべての CRC を示すリストが表示されます。

- ・ [概念]：ソース内の概念が降順に示されます。
- ・ CRC：ソース内の CRC が概念と関係を示す強調表示付き（上記参照）で降順に示されます。[CRC] のリストには、無関係な単語やパス関係の単語は含まれず、属性は示されません。

[インデックス作成結果] ウィンドウの右上にある[次でソート] ボタンを使用すると、[概念] と [CRC] のリストを切り替えて、頻度のカウントまたは優位性の値を降順で表示できます。

[概念] のリストでは、最も優位性の高い概念に、優位性の値 1000 が割り当てられます。優位性の低い概念には、小さい値が割り当てられます。ソースが大きくなると、優位性が最低の値が小さくなる傾向があります。例えば、25 個の概念

が含まれているソースでは、優位性の範囲が 1000 から 83、300 個の概念が含まれているソースでは、優位性の範囲が 1000 から 2 のようになります。

[CRC] のリストでは、優位性スコアは、概念と関係の優位性の値を加算したものになります。

注釈 ドメインでサポートされる唯一の言語が日本語の場合、[インデックス作成結果] には、[概念] と [CRC] のリストの代わりに、1 つの [エンティティ] のリストが表示されます。

4

REST インタフェース

InterSystems IRIS® Data Platform の **%iKnow.REST.v1** クラスの一般的な NLP タスクに REST API インタフェースが提供されます。この REST API は、基礎 (ドメインと構成のクエリ)、従来型のクエリ (セマンティックを含む)、シンプルなドメイン処理 (ソースの追加/削除、skiplist 管理)、およびマッチング API とディクショナリ API に対するサポートを扱います。

この API の主な属性のいくつかを以下に示します。

- ・ ほとんどの API 呼び出しには GET と POST のバリエーションがあります。基本ではないすべてのパラメータに対する GET バリエーションは既定の値となり、最も簡単に呼び出せます。POST バリエーションは、Request オブジェクトとして渡された JSON 文字列を使用してすべての関連引数へのアクセスを提供します。これは、URL に入力する内容を補完し、重複している場合にはこの内容を上書きします。例えば、スラッシュを含む一部のエンティティ値は、GET 要求では使用できませんが、POST 要求で送信される JSON プロパティとして指定できます。
- ・ POST 要求に対して使用可能な引数は **%iKnow.REST.v1** のクラス・リファレンスで、いくつかの共有引数 (フィルタと強調表示) は **%iKnow.REST.Base** で簡潔に説明されています。
- ・ URL は全体を必ず小文字で表記します。POST 引数と返された JSON プロパティ名は、キャメルケースで表記します。ただし、最初の文字は小文字になります。
- ・ 返された JSON は一般的に包括的なものであり、対応するフラット (SQL と互換性のある) クエリで提供されるものよりもはるかに多くの情報を返します。例えば、エンティティを含むすべてのソースを要求すると、ソース、メタデータ、および必要に応じて強調表示されたスニペットのリストが返されます。パフォーマンス上の理由で、この機能のいくつかを除外することができますが、既定では、ほとんどの機能が自動的に含まれます。

4.1 Swagger

NLP REST API のドキュメントは、[OpenAPI Specification](#) (Swagger と呼ばれます) を使用して完全に作成されます。YAML の記述には `/swagger` エンドポイントからアクセスでき、この API の最上部で便利な GUI 機能を提供する [swagger-ui](#) へ直接ロードできます。

それを使用するには、swagger-ui をインストールするか、<http://petstore.swagger.io> に移動して、このエンドポイントを指します。

- ・ swagger-ui をダウンロードする
- ・ swagger-ui ダウンロードを unzip します。
- ・ Swagger ボックスに以下のどちらかを指定します。
 - 最小のセキュリティの InterSystems IRIS : `http://localhost:57775/api/iknow/v1/samples/swagger` (57775 を Web サーバのポート番号に、samples を必要なネームスペースに置き換えます)。

- 通常のセキュリティの InterSystems IRIS :
`http://localhost:57775/api/iknow/v1/samples/swagger?IRISUsername=myname&IRISPassword=mypassword`
(57775 を Web サーバのポート番号に、myname を InterSystems IRIS ユーザ名に、samples を必要なネームスペースに、mypassword を InterSystems IRIS アカウントのパスワードに置き換えます)。
- [エクスプローラ] ボタンを選択します。

4.1.1 Swagger を使用して NLP データを返す

各 REST 文の Swagger-UI 表示を使用して、REST API を直接呼び出すか、ブラウザ・ウィンドウで REST API を呼び出すために使用できる要求 URL 値を提供できます (GET 要求用)。

REST API を直接呼び出すには、必須 **パラメータ** を指定します。一般的に、NLP REST API には、整数として指定されるドメイン ID パラメータが必要です。例えば、2 という ID を持つドメインの詳細を GET すると、以下の要求 URL が生成されます。

http://localhost:57775/api/iknow/v1/user/domain/2/details

Swagger インタフェースは、/swagger エンドポイントにアクセスするために使用した URL から使用するネームスペースを推測することに注意してください。例えば、ネームスペース SAMPLES で REST API を呼び出すには、以下の URL を使用して swagger エンドポイントをロードします。

<http://localhost:57775/api/iknow/v1/samples/swagger>

NLP データが JSON 形式でブラウザに返されます。例えば、この要求 URL は "crashes" という名前のドメインに対して以下の JSON 文字列を返します。

[illegible]

4.2 REST 操作

NLP REST 操作は以下のトピックに分類されます。

- ・ ドメインと構成の情報 (“Miscellaneous” の下にある Swagger にリスト)
- ・ 指定のドメイン内のソース。
- ・ 指定のドメイン内のエンティティ。
- ・ 指定のドメイン内の文。
- ・ 指定のドメイン内のパスと CRC。
- ・ 指定のドメイン内のディクショナリとマッチング。
- ・ 指定のドメイン内の skiplist。
- ・ 強調表示 (“Miscellaneous” の下にある Swagger にリスト)

コア REST API は (ほぼすべての URL の開始時にドメイン ID を指定することで) あらゆる定義済みドメインへのアクセスをサポートしますが、ドメイン固有の REST API を生成して、webapp を通じて単一ドメインを公開することもできます。そのためには、`%Know.REST.Base` クラスにある `CreateDomainAPI()` メソッドを使用して、新しい Web アプリケーションを作成し、新しく作成したクラスを「ディスパッチ・クラス」と呼びます。これは、特定のドメインへのアクセスのみを有効にし、一般的な `/api/iknow/` Web アプリケーションを無効にするときに役に立ちます。

これらの REST 操作について、指定のドメインが存在しない場合、特に断りがない限り、その操作では

```
{
  "errors": [
    {
      "error": "ERROR #8021: Domain with id 7 does not"
    }
  ]
}
```


exist", "code": 8021, "domain": "%ObjectErrors", "id": "IKNoDomainWithId", "params": [{"7"}]}, "summary": "ERROR #8021: Domain with id 7 does not exist"} が返されます。

4.2.1 ドメインと構成

ドメインと構成に対する NLP REST 操作 (“miscellaneous” のカテゴリに分類) は以下ようになります。

- ・ [ドメインのリスト] : 現在のネームスペースですべてのドメインに関する情報を取得するための GET と POST。GetDomains() を参照してください。

定義済みドメインがない場合、{"domains": []} を返します。それ以外の場合は、ドメイン名の順でドメインの JSON 配列を返します。各ドメインについて、ドメイン ID、ドメイン名、NLP バージョン、definitionClass 名、および sourceCount がリストされます。

ドメインのリストは、このガイドの “NLP 環境” の章に詳しく説明されています。

- ・ [リストの構成] : 現在のネームスペースですべての構成に関する情報を取得するための GET と POST。GetConfigurations() を参照してください。

NLP に対応したネームスペースには構成 {"configurations": [{"id": 1, "name": "DEFAULT", "languages": ["en"]}]} が割り当てられます。追加の構成を明示的に定義することができます。ドメイン・アーキテクトを使用してドメインを作成すると、NLP は対応する構成を作成します。

このオプションは、構成名の順で構成の JSON 配列を返します。各構成について、構成 ID、構成名、サポートされている言語の JSON 配列、および (オプションで) userDictionary JSON オブジェクトがリストされます。言語は既定で英語 (["en"]) に設定されます。

構成のリストは、このガイドの “NLP 環境” の章に詳しく説明されています。

- ・ [ドメインの詳細] : 指定のドメインに関する詳細情報を取得するための GET と POST。GetDomainDetails() を参照してください。

返されるデータには、ドメインの ID、名前、パラメータ、およびメタデータが含まれます。

- パラメータには、DefaultConfig (domainname.Configuration)、DefinitionClass、および ManagedBy (既定は DefinitionClass) が含まれます。
- メタデータには、メタデータ ID、メタデータ名 (既定は DateIndexed)、サポートされている演算子の配列、データ型 (DateIndexed はデータ型日付)、ストレージ (既定は通常)、caseSensitive (ブーリアン値)、および非表示 (ブーリアン値) が含まれます。

4.2.2 ソース

ソースに対する NLP REST 操作は以下ようになります。

- ・ [フィルタ処理されたソース] : 指定されたフィルタを渡すすべてのソースを取得するための GET と POST。既定では、すべてのソースを返します。GetSources() を参照してください。

ドメインは存在するものの、フィルタを渡すソースが含まれていない場合は、{"sources": []} を返します。ソースがある場合は、ID、extId (外部 ID)、メタデータ (ソースのインデックスが作成された日付と時間)、および各ソースのテキスト断片を返します。この断片はテキストからの 2 つの文で構成されます。それらの文が連続するものではない場合、省略記号 (3 つのドット) によって連続していない部分が示されます。テキストに含まれる文が 2 つ以下の場合、この断片にはフルテキストが含まれます。

ソースのフィルタ処理は、このガイドの “ソースのフィルタ処理” の章に詳しく説明されています。

- ・ [エンティティ別のソース] : 指定のエンティティを含むソースを取得するための GET と POST。GetSourcesByEntity() を参照してください。

ドメインは存在するものの、フィルタを渡すソースが含まれていない場合は、{"sources":[]} を返します。ソースがある場合は、ID、extId (外部 ID)、メタデータ (ソースのインデックスが作成された日付と時間)、および各ソースのテキスト断片を返します。この断片は、指定のエンティティが含まれるテキストからの (多くても) 2 つの文で構成されます。これらの文は必ずしも順番どおりに表示されるとは限りません。

[エンティティの一致によるソースのフィルタ処理](#)は、このガイドの“ソースのフィルタ処理”の章に詳しく説明されています。

- ・ [CRC 別のソース]: 指定された CRC を含むソースを取得するための GET と POST。これらの操作を使用して、CRC (concept:relation:concept)、CR (concept:relation)、RC (:relation:concept)、または CC (concept::concept) を取得することができます。GetSourcesByCRC() を参照してください。

ドメインは存在するものの、CRC を含むソースが含まれていない場合は、{"sources":[]} を返します。CRC を含むソースがある場合は、ID、extId (外部 ID)、メタデータ (ソースのインデックスが作成された日付と時間)、および各ソースのテキスト断片を返します。この断片は、指定された CRC が含まれるテキストからの (多くても) 2 つの文で構成されます。これらの文は必ずしも順番どおりに表示されるとは限りません。

CR の後には概念が必ず続きます。そのため、CR pilot:stated は "pilot stated the airplane"、"pilot stated he"、および "pilot stated ¥"I then" を返します。"pilot stated that" や "pilot stated to" は返しませんが。

- ・ [類似ソース]: 指定されたソースに類似するソースを取得するための GET と POST。GetSimilarSources() を参照してください。

ドメインは存在するものの、類似ソースが含まれていない場合は、{"sources":[]} を返します。[類似ソース](#)がある場合、類似性スコアによる降順でエンティティの JSON 配列を返します。同じ類似性スコアを持つ相似した複数のエンティティがある場合は、ID 順でリストされます。各類似ソースについて、ID、extId (外部 ID)、スコア (類似性スコア)、percentageMatched、percentageNew、numberOfEntitiesInRefSource、numberOfEntitiesInCommon、numberOfEntitiesInThisSource、metadata (ソースのインデックスが作成された日付と時間)、および各ソースのテキスト断片を返します。この断片は、類似エンティティが含まれるテキストからの (多くても) 2 つの文で構成されます。

[類似ソース](#)は、このガイドの“NLP クエリ”の章に詳しく説明されています。

- ・ [ソースの詳細]: 指定されたソースに関する詳細を取得するための GET と POST。GetSourceDetails() を参照してください。

指定されたソースについて、仮想ブーリアン、メタデータ (ソースのインデックスが作成された日付と時間)、およびソースのフルテキストを返します。

- ・ [ソースの追加]: ドメインにソースを追加するための POST。AddSource() を参照してください。

[ソースの追加](#)は、このガイドの“プログラムによるテキスト・データのロード”の章に詳しく説明されています。

- ・ [ソースの削除]: ドメインからソースを削除するための DELETE。DropSource() を参照してください。

[ソースの削除](#)は、このガイドの“プログラムによるテキスト・データのロード”の章に詳しく説明されています。

- ・ [ソースのメタデータ]: 特定のソースのメタデータを更新するための POST。SetMetadata() を参照してください。

4.2.3 エンティティ

エンティティに対する NLP REST 操作は以下のようになります。

- ・ [上位エンティティ]: 指定のドメインで上位エンティティを取得するための GET と POST。GetEntities() を参照してください。

エンティティを含む定義済みドメインがない場合、{"entities":[]} を返します。エンティティがある場合、周期による降順でエンティティの JSON 配列を返します。同じ周期を持つ複数のエンティティがある場合は、ID 順でリストされます。各エンティティは、ID、値、周期、およびスプレッドのキー/値ペアを持つ JSON オブジェクトとしてリストされます。

[上位エンティティ](#)は、このガイドの“NLP クエリ”の章に詳しく説明されています。

- ・ [類似エンティティ]: 指定した文字列に類似するエンティティを取得するための GET と POST。GetSimilarEntities() を参照してください。

類似エンティティがない場合、{"entities":[]} を返します。類似エンティティがある場合、周期による降順でエンティティの JSON 配列を返します。同じ周期を持つ相似した複数のエンティティがある場合は、ID 順でリストされます。各エンティティは、ID、値、周期、およびスプレッドのキー/値ペアを持つ JSON オブジェクトとしてリストされます。

類似エンティティは、このガイドの“NLP クエリ”の章に詳しく説明されています。

- ・ [関連するエンティティ]: 指定エンティティに関連するエンティティを取得するための GET と POST。GetRelatedEntities() を参照してください。

関連するエンティティがない場合、{"entities":[]} を返します。関連するエンティティがある場合、近似による降順でエンティティの JSON 配列を返します。同じ近似を持つ相似した複数のエンティティがある場合は、ID 順でリストされます。各エンティティは、ID、値、および近似のキー/値ペアを持つ JSON オブジェクトとしてリストされます。

関連するエンティティは、このガイドの“NLP クエリ”の章に詳しく説明されています。

- ・ [エンティティの詳細]: 指定されたエンティティに関する詳細を取得するための GET と POST。GetEntityDetails() を参照してください。

指定のエンティティがドメインに存在しない場合、または指定のドメインが存在しない場合、{"id":"","value":"armadillo"} を返します。エンティティが見つかった場合は、ID、値のキー/値ペアを持つ JSON オブジェクト、および metricsAsConcept、metricsAsRelation、metricsAsAny の各サブ配列を返します。各サブ配列では、周期、スプレッド、および tfidf がリストされます。

エンティティが概念としてのみ発生する場合、metricsAsRelation の値は 0 になり、metricsAsAny の値は metricsAsConcept の値と同じになります。エンティティが関係としてのみ発生する場合、metricsAsRelation の値は 0 になり、metricsAsAny の値は metricsAsRelation の値と同じになります。エンティティが概念としても関係としても発生する場合、周期とスプレッドに対する metricsAsAny の値は、それぞれの概念の値と関係の値の合計になります。tfidf の値は、対応する概念の値と関係の値から算出されます。

4.2.4 文

文に対する NLP REST 操作は以下のようになります。

- ・ [エンティティ別の文]: 指定のエンティティを含む文を取得するための GET と POST。GetSentencesByEntity() を参照してください。

指定のエンティティを含む文がない場合、{"sentences":[]} を返します。エンティティを含む文がある場合、文の JSON 配列を返します。各文について、文 ID、ソース ID、文のテキスト、および文の部分でリストします。部分は文の順にリストされます。各部分は部分配列の要素です。部分配列の各部分について、partId、リテラル (大文字/小文字の区別は元のまま)、ルール (概念、関係、pathRelevant、または nonRelevant)、entityId、およびエンティティ (小文字) をリストします。部分が nonRelevant の場合、entityId やエンティティはリストされません。

- ・ [文の詳細]: 指定された文に関する詳細を取得するための GET と POST。GetSentenceDetails() を参照してください。

指定された文が存在する場合、文 ID、文の部分、テキスト、および sourceId を返します。部分は文の順にリストされます。各部分は部分配列の要素です。部分配列の各部分について、partId、リテラル (大文字/小文字の区別は元のまま)、ルール (概念、関係、pathRelevant、または nonRelevant)、entityId、およびエンティティ (小文字) をリストします。部分が nonRelevant の場合、entityId やエンティティはリストされません。

4.2.5 パスと CRC

パスと CRC (概念 - 関係 - 概念の文字列) に対する NLP REST 操作は以下のようになります。

- ・ [上位のCRC]: 指定のドメインで上位の CRC を取得するための GET と POST。GetCRCs() を参照してください。

CRC を含むソースがない場合、{"crcs":[]} を返します。周期による降順で CRC の JSON 配列を返します。同じ周期を持つ複数の CRC がある場合は、ID 順でリストされます。各 CRC について、id、ヘッド概念、関係、テール概念、周期、およびスプレッドをリストします。一部の CRC では、ヘッドまたはテールの値は "" (空の文字列) になります。例えば、ヘッドのキーと値のペアは、CRC で関係 "according to" または "during" で始まる値 "" を持ちます。この CRC のヘッドは指定されていないためです。テールのキーと値のペアは、CRC で "occurred" または "said" のような関係を含む値 "" を持ちます。この CRC のテールは指定されていないためです。

- ・ [エンティティ別の CRC]: 指定のエンティティを含む CRC を取得するための GET と POST。GetCRCsByEntity() を参照してください。

指定のエンティティを含む CRC を持つソースがない場合、{"crcs":[]} を返します。その他の場合は、周期による降順で CRC の JSON 配列を返します。同じ周期を持つ複数の CRC がある場合は、ID 順でリストされます。各 CRC について、id、ヘッド概念、関係、テール概念、周期、およびスプレッドをリストします。一部の CRC では、ヘッドまたはテールの key:value ペアは省略されます。

- ・ [エンティティ別のパス]: 指定のエンティティを含むパスを取得するための GET と POST。GetPathsByEntity() を参照してください。

指定のエンティティを含むパスを持つソースがない場合、{"paths":[]} を返します。その他の場合は、パスの JSON 配列を返します。各パスは、パス ID、エンティティ配列、sourceId、および sentenceId で構成されます。エンティティ配列には、各エンティティのエンティティ ID、partId、エンティティ値、ロール指定子 (概念、関係、または [pathRelevant](#))、および (オプションで) 属性配列が含まれます。属性配列は、タイプ (否定演算子)、レベル (パス、語)、またレベルが語の場合は key:value ペアの語で構成されます。例えば、

ば、"attributes": [{"type": "negation", "level": "path"}]。

4.2.6 デクショナリとマッチング

[デクショナリ](#)に対する NLP REST 操作は以下のようになります。

- ・ [デクショナリの作成]: デクショナリを新規作成するための POST。CreateDictionary() を参照してください。
[デクショナリの作成](#)は、このガイドの“スマート・マッチング: デクショナリの作成”の章に詳しく説明されています。
- ・ [デクショナリ (複数) の取得]: 指定のドメインに対して定義されたデクショナリを取得するための GET と POST。GetDictionaries() を参照してください。
定義済みデクショナリがない場合、{"dictionaries":[]} を返します。
- ・ [デクショナリの取得]: 指定のデクショナリの詳細な内容を取得するための GET と POST。GetDictionaryDetails() を参照してください。
- ・ [デクショナリの削除]: 指定のデクショナリを削除するための DELETE。DropDictionary() を参照してください。
- ・ [項目の追加]: 1 つ以上の項目をデクショナリに追加するための POST。CreateDictionaryItems() を参照してください。
- ・ [項目の詳細]: デクショナリ項目の詳細な内容を取得するための GET と POST。GetDictionaryItemDetails() を参照してください。
- ・ [項目の削除]: 指定のデクショナリ項目を削除するための DELETE。DropDictionaryItem() を参照してください。
- ・ [用語の追加]: 1 つ以上の用語をデクショナリ項目に追加するための POST。CreateDictionaryTerms() を参照してください。
- ・ [用語の削除]: デクショナリ項目から指定された用語を削除するための DELETE。DropDictionaryTerm() を参照してください。

[デクショナリ](#)を使用したマッチング処理のための NLP REST 操作は以下のようになります。

- ・ [ソース (複数) のマッチング]: ディクショナリに対してすべてのソースをマッチングさせるための GET と POST。MatchAll() を参照してください。
ディクショナリの一致によるソースのフィルタ処理は、このガイドの“ソースのフィルタ処理”の章に詳しく説明されています。
- ・ [ソースのマッチング]: ディクショナリに対して指定のソースをマッチングさせるための GET と POST。GetMatchesBySource() を参照してください。
- ・ [ディクショナリの一致]: 指定されたソースについてディクショナリの一致をすべて取得するための GET と POST。複数のディクショナリを指定できます。MatchSource() を参照してください。
- ・ [項目の一致]: 指定されたディクショナリ項目に対するすべての一致を取得するための GET と POST。GetMatchesByItem() を参照してください。

4.2.7 Skiplist

skiplist に対する NLP REST 操作は以下のようになります。

- ・ [skiplist の一覧]: ドメインの全 skiplist の一覧を取得するための GET と POST。“GetSkiplists()”を参照してください。
定義済み skiplist がいない場合、{"skiplists":[]} を返します。1 つ以上の skiplist がある場合、skiplist の JSON 配列を返します。各 skiplist は、ID、名前に対するキー/値ペアを持つ JSON オブジェクトと、要素の JSON 配列としてリストされます。
- ・ [skiplist の作成]: skiplist を新規作成するための GET と POST。“CreateSkiplist()”を参照してください。
skiplist の作成については、このガイドの“Skiplist”の章に詳しく説明されています。
- ・ [skiplist の取得]: 指定された skiplist の内容を取得するための GET と POST。“GetSkiplistDetails()”を参照してください。
ID、名前に対するキー/値のペアを持つ JSON オブジェクトと、要素の JSON 配列として、skiplist ID で指定される skiplist を取得します。
- ・ [skiplist の削除]: 指定された skiplist を削除するための DELETE。“DropSkiplist()”を参照してください。
- ・ [skiplist のクリア]: 指定された skiplist の内容をすべてクリア (削除) するための GET と POST。“ClearSkiplist()”を参照してください。
- ・ [エンティティの追加]: 指定された skiplist にエンティティを追加するための GET と POST。“AddStringToSkiplist()”を参照してください。
- ・ [エンティティの削除]: 指定された skiplist からエンティティを削除するための GET と POST。“RemoveStringFromSkiplist()”を参照してください。

5

手動による NLP 環境の作成

ドメイン・アーキテクトを使用して InterSystems NLP ドメインを作成または編集するときは、InterSystems NLP によって 3 つのオブジェクトのインスタンスが自動的に作成されます。これら 3 つのオブジェクトをまとめて、テキスト・ソースをロードする環境を定義します。これらのオブジェクトは以下のとおりです。

- ・ **ドメイン**：InterSystems NLP によるオペレーションで使用する論理空間を確立します。ドメインの指定は必須です。
- ・ **構成**：ソース・ドキュメント・コンテンツのための言語環境を確立します。構成の指定はオプションです。ドメインを指定しない場合のために、InterSystems NLP には既定値が用意されています。ドメイン・アーキテクトでは、ドメインの言語環境を定義できます。この章では、追加の機能およびオプションと共に、ドメイン非依存の構成を作成する方法について説明します。
- ・ **ユーザ・ディクショナリ**：ソース・テキストを InterSystems NLP にロードするときに適用する代入と置き換えのカスタム・セットを確立します。ユーザ・ディクショナリは、例えば**意味的属性マーカ**やフィールド固有のコンセプトとして文字列を特定することで、エンジンの既定の動作を拡張します。1 つ以上の構成にユーザ・ディクショナリが提供されます。ユーザ・ディクショナリの指定は任意です。指定しない場合、ユーザ・ディクショナリは使用されません。

これらのオブジェクトのインスタンスを手動で作成することで InterSystems NLP 環境を作成することもできます。ドメイン、構成、およびユーザ・ディクショナリの複数のインスタンスを作成できます。これらの環境オブジェクトは互いに依存しておらず、特定セットのソース・データに依存することはありません。

このページでは、ドメイン、構成、およびユーザ・ディクショナリのオブジェクトについて詳しく説明し、これらのオブジェクトを使用して InterSystems NLP 環境を手動で作成する方法を説明します。

5.1 NLP ドメイン

InterSystems NLP のすべてのオペレーションはドメイン内部で発生します。ドメインは、InterSystems IRIS® データ・プラットフォームのネームスペースの中で InterSystems NLP によって定義したユニット (単位) です。InterSystems NLP が使用するすべてのソース・データは、ドメインにリストされ、ロードされます。1 つのネームスペースに複数のドメインを設定できます。

注釈 **ネームスペースの作成**時にそれが別のネームスペースからコピーされた場合、それらのネームスペース間でドメインが共有されます。ネームスペースを作成するときには、ドメインがそのネームスペースで一意になるように、**グローバルの新しいデータベースを作成**する必要があります。

InterSystems NLP ドメインは以下の 3 つの方法で、定義、変更および削除できます。

- ・ **ドメイン・アーキテクトを使用したドメインの定義**。これは、ドメインを定義して、そのドメインの言語、メタデータ、およびロードするデータを指定するための最も簡単な方法です。

- ・ %iKnow.DomainDefinition のサブクラスとしてのドメインの定義。このオプションは、ドメインを定義して、そのドメインの構成設定、メタデータ、およびロードするデータを指定するための強力な機能を備えた方法になります。
- ・ %iKnow.Domain クラス・メソッドおよびプロパティを使用したプログラムによるドメインの定義。

5.1.1 DomainDefinition を使用したドメインの定義

%iKnow.DomainDefinition から継承したクラスを作成およびコンパイルすると、そのクラスのドメイン XData ブロックで XML 表現によって指定した設定に対応する InterSystems NLP ドメインがコンパイラによって自動的に作成されます。ユーザは、ドメイン・パラメータ、メタデータ・フィールド定義、割り当て済みの構成などの静的要素を指定できます。これらはすべてコンパイル時に自動的に作成されます。また、ユーザは、ドメインにロードするテキスト・データのソースを指定することも可能です。InterSystems IRIS は、このソース情報を使用して、[classname].Domain という新しいクラスに専用の %Build() メソッドを生成します。この %Build() メソッドを使用して、指定されたデータをこのドメインにロードできます。

以下の手順を使用して、%iKnow.DomainDefinition からの継承により、ドメインを定義します。

1. スタジオにて、目的のネームスペースを選択してから ([ファイル]→[ネームスペース変更])、新しいクラス定義を作成します ([ファイル]→[新規]、[一般] タブから [クラス定義] アイコンを選択)。[新規クラスウィザード] が起動します。
2. [新規クラスウィザード] にて、任意のパッケージ名とクラス名を指定します。[次へ] を押します。[クラスタイプ] ボックスにて、[Extends] を選択して、スーパークラスの名前として %iKnow.DomainDefinition を指定します。[完了] を押します。
3. スタジオにて、[クラス]→[リファクタ]→[オーバーライド] を選択します。[XData] タブを選択します。[ドメイン] アイコンを選択します。[OK] を押します。これにより、XData ブロックが作成されます。
4. XData ブロックの中括弧内にカーソルを置いて、“<” を入力します。スタジオ・アシストが即座に XML コード・オプションの提供を開始します。任意のドメイン名を指定します。少なくとも、以下の指定が必要となります。

```
{
  <domain name="AviationEvents">
  </domain>
}
```

XML 構文とスタジオ・アシストのオプションを使用すれば、他のプロパティを XData ブロック内に追加できます (下記で説明)。少なくとも、<data> </data> タグが必要で、これらの中にデータ・ソースが必要です。例えば、以下のよう指定します。

```
{
  <domain name="AviationEvents">
    <data>
      <files path="C:\MyDocs\" />
    </data>
  </domain>
}
```

5. スタジオにて、ファイルを保存してから (この場合は “MyDomain“)、[ビルド]→[コンパイル] を選択します。これによりドメインが作成されます。

以下に、この種のドメイン定義の例を示します。

Class Definition

```
Class Aviation.MyDomain Extends %iKnow.DomainDefinition
{
  /// An XML representation of the domain this class defines.
  XData Domain [ XMLNamespace = "http://www.intersystems.com/iknow" ]
  {
    <domain name="AviationEvents">
      <parameter name="Status" value="1" />
      <configuration name="MyConfig" detectLanguage="1" languages="en,es" />
      <parameter name="DefaultConfig" value="MyConfig" />
      <metadata>
        <field name="EventDate" dataType="DATE" />
      </metadata>
    </domain>
  }
}
```

```

        <field name="Type" dataType="STRING" />
    </metadata>
    <data dropBeforeBuild="true">
        <files path="C:\MyDocs\" encoding="utf-8" recursive="1" extensions="txt"
            configuration="MyConfig" />
        <query sql="SELECT %ID,EventDate,Type,NarrativeFull
            FROM Aviation.Event
            idField="ID" groupField="ID" dataFields="NarrativeFull"
            metadataFields="EventDate,Type" metadataColumns="EventDate,Type" />
    </data>
</domain>
}

```

コンパイル時、この定義はドメイン "AviationEvents" (Status パラメータを 1 に設定) および 2 つのメタデータ・フィールドを作成します。これは、英語 (en) およびスペイン語 (es) のテキストを処理する構成 "MyConfig" を定義し、上記のドメインに割り当てます。

この定義は、ロードするファイルをこのドメインに指定します。テキスト (.txt) ファイルが **C:¥MyDocs¥** ディレクトリからロードされ、InterSystems SQL データが Aviation.Event テーブルからロードされます。詳細は、**%iKnow.Model.listFiles** クラス・プロパティおよび **%iKnow.Model.listQuery** クラス・プロパティを参照してください。

依存クラス **Aviation.MyDomain.Domain** の **%Build()** メソッドが生成されます。このメソッドには、**C:¥MyDocs** ディレクトリおよび Aviation.Event テーブルからデータをロードするロジックがあります。

指定したテキスト・データ・ソースをドメインにロードするには、以下のように行います。

ObjectScript

```
SET stat=##class(Aviation.MyDomain).%Build()
```

%Build() を使用後には、以下のようにエラーをチェックできます。

ObjectScript

```
DO $SYSTEM.iKnow.ListErrors("AviationEvents",0)
```

これにより、3 タイプのエラー (エラー、障害のあるソース、警告) がリストされます。

現在のネームスペースで定義されたドメイン、および各ドメインに対してロードされたソースの数を表示するには、以下のようになります。

ObjectScript

```
DO $SYSTEM.iKnow.ListDomains()
```

このドメインに対して定義されたメタデータ・フィールドを表示するには、以下のようになります。

ObjectScript

```
DO $SYSTEM.iKnow.ListMetadata("AviationEvents")
```

InterSystems NLP では、すべてのドメインに 1 つのメタデータ・フィールド (DateIndexed) を割り当てます。追加のメタデータ・フィールドも定義することができます。

このドメイン定義で、**<matching>** 要素を指定できます。**<matching>** 要素は、ディクショナリ情報を記述し、ロードしたソースをこれらのディクショナリに自動的にマッチングするかどうかを指定します。InterSystems IRIS は、クラスのコンパイル中にこれらのオブジェクトに関して基本的な検証を実行しますが、これらのオブジェクトが **%Build()** メソッドの一部としてロードされるため、実行時にのみ名前の重複が発生します。詳細は、**%iKnow.Model.matching** クラス・プロパティを参照してください。

このドメイン定義で、**<metrics>** 要素を指定できます。**<metrics>** 要素は、カスタム・メトリックをドメインに追加します。**%iKnow.Metrics.MetricDefinition.Register()** を呼び出す必要はありません。これは、コンパイル時にドメイン定義のコードで自動的に実行されます。詳細は、**%iKnow.Model.metrics** クラス・プロパティを参照してください。

5.1.2 プログラムによるドメインの定義

クラス・メソッドを使用して新規ドメインを定義するには、%iKnow.Domain.%New() 永続メソッドを呼び出して、ドメイン名をメソッド・パラメータとして指定します。ドメイン名には、任意の有効な文字列を指定できます。ドメイン名では、大文字と小文字は区別されません。このドメインに割り当てる名前は、現在のネームスペースで一意的でなければなりません。このメソッドは、InterSystems IRIS インスタンスの全ネームスペースで一意的であるドメイン・オブジェクト参照 (oref) を返します。%Save() メソッドを使用して、このインスタンスを永続的に保存する必要があります。ドメインの ID プロパティ (整数値) は、以下の例で示すように、インスタンスを永続オブジェクトとして保存するまで定義されません。

ObjectScript

```
CreateDomain
SET domOref=##class(%iKnow.Domain).%New("FirstExampleDomain")
WRITE "Id before save: ",domOref.Id,!
DO domOref.%Save()
WRITE "Id after save: ",domOref.Id,!
CleanUp
DO ##class(%iKnow.Domain).%DeleteId(domOref.Id)
WRITE "All done"
```

NameIndexExists() を使用して、ドメインが既に存在しているかどうかを確認します。ドメインが存在する場合、NameIndexOpen() を使用して、そのドメインを開きます。ドメインが存在しない場合は、%New() を使用してドメインを作成した後、%Save() を使用します。

以下の例では、ドメインが存在するかどうかをチェックします。ドメインが存在しない場合は、プログラムによって作成されます。ドメインが存在する場合は、それが開かれます。デモンストレーションのため、このプログラムでは、ランダムにドメインが削除されたり削除されなかったりします。

ObjectScript

```
DomainCreateOrOpen
SET domn="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ WRITE "The ",domn," domain already exists",!
  SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
  SET domId=domo.Id
}
ELSE {
  SET domo=##class(%iKnow.Domain).%New(domn)
  DO domo.%Save()
  SET domId=domo.Id
  WRITE "Created the ",domn," domain",!
  WRITE "with domain ID ",domId,! }
ContainsData
SET x=domo.IsEmpty()
IF x=1 {WRITE "Domain ",domn," contains no data",!}
ELSE {WRITE "Domain ",domn," contains data",!}
CleanupForNextTime
SET rnd=$RANDOM(2)
IF rnd {
  SET stat=##class(%iKnow.Domain).%DeleteId(domId)
  IF stat {WRITE "Deleted the ",domn," domain" }
  ELSE { WRITE "Domain delete error:",stat }
}
ELSE {WRITE "No delete this time" }
```

ドメインを作成または開く %iKnow.Domain クラス・メソッドは、出力 %Status パラメータと共に提供されます。このパラメータは、現在のシステムに InterSystems NLP へのライセンス・アクセスがないために、InterSystems NLP ドメインを作成することや開くことができない場合に設定されます。

5.1.3 ドメイン・パラメータの設定

さまざまな InterSystems NLP オペレーションの動作をドメイン・パラメータで制御します。個々のパラメータは、該当する場所に記載されています。使用可能なすべてのドメイン・パラメータは、付録“[ドメイン・パラメータ](#)”を参照してください。

注釈 以下の例のドメイン・パラメータは、パラメータ名 (FullMatchOnly など) ではなく、マクロ・パラメータ (\$\$IKP-FULLMATCHONLY など) によって参照されます。プログラミングの方法としては、パラメータ名ではなく、このような %IKPublic マクロを使用することをお勧めします。

すべてのドメイン・パラメータに既定値が取得されています。一般的に、InterSystems NLP では、任意のドメイン・パラメータを特に設定しなくても最適な結果が得られます。InterSystems NLP では以下のように各パラメータの値が決まります。

1. 現在のドメインのパラメータ値を指定している場合は、その値が使用されます。データをドメインにロードする前にのみ設定可能なパラメータと、常時設定可能なパラメータがあることに注意します。IsEmpty() メソッドを使用することで、データが現在のドメインにロードされているか判断できます。
2. システム全体のパラメータ値を指定している場合は、その値がすべてのドメインの既定値として使用されますが、ドメイン固有の値が設定されているドメインはその対象外となります。
3. ドメイン・レベルでもシステム・レベルでもパラメータの値を指定していない場合、InterSystems NLP ではそのパラメータの既定値が使用されます。

5.1.3.1 現在のドメインのパラメータ設定

ドメインを作成すると、SetParameter() インスタンス・メソッドを使用して、個々のドメインのドメイン・パラメータを設定できます。SetParameter() によって、指定したパラメータが有効で設定されていたことを示すステータスが返されます。GetParameter() によって、パラメータ値とそのパラメータが設定されていたレベル (DEFAULT、DOMAIN または SYSTEM) が返されます。GetParameter() は、パラメータ名の有効性をチェックすることなく、ドメイン・レベルまたはシステム・レベルで設定されている際に識別できないすべてのパラメータ名に対して DEFAULT を返すことに注意します。

以下の例では、SortField **ドメイン・パラメータ**の既定値を取得し、現在のドメインにそのパラメータを設定してから、設定した値と設定された時点でのレベル (DOMAIN) を取得します。

ObjectScript

```
#include %IKPublic
DomainCreate
    SET domn="paramdomain"
    SET domo=##class(%iKnow.Domain).%New(domn)
    WRITE "Created the ",domn," domain",!
    DO domo.%Save()
DomainParameters
    SET sfval=domo.GetParameter($$IKPSORTFIELD,.sf)
    WRITE "SortField before SET=",sfval," ",sf,!
    IF sfval=0 {WRITE "changing SortByFrequency to SortBySpread",!
        SET stat=domo.SetParameter($$IKPSORTFIELD,1)
        IF stat=0 {WRITE "SetParameter failed" QUIT} }
    WRITE "SortField after SET=",domo.GetParameter($$IKPSORTFIELD,.str)," ",str,!
CleanupForNextTime
    SET stat=##class(%iKnow.Domain).%DeleteId(domo.Id)
    IF stat {WRITE "Deleted the ",domn," domain" }
    ELSE { WRITE "Domain delete error:",stat }
```

5.1.3.2 システム全体のパラメータの設定

SetSystemParameter() メソッドを使用して、すべてのドメインのドメイン・パラメータをシステム全体で設定できます。このメソッドを使用して設定したパラメータは、その時点で、すべてのネームスペースにおける既存およびそれ以降に作成されるすべてのドメインの既定パラメータ値となります。このシステム全体の既定値は、SetParameter() インスタンス・メソッドを使用して、個々のドメインについてオーバーライドされます。

注釈 **SortField** および **Jobs** のドメイン・パラメータは例外となっています。これらのパラメータをシステム・レベルで設定しても、ドメイン設定に影響を与えることはありません。

ドメイン・パラメータがシステムの既定として設定されているかどうかを確認するには、GetSystemParameter() メソッドを使用できます。システム全体のパラメータの初期値は常に NULL 文字列です (既定値なし)。

ドメイン・パラメータのシステム全体の既定の設定を削除する場合は、UnsetSystemParameter() メソッドを使用します。システム全体のパラメータ設定が設定された後、新しい値に設定するには、まず設定を解除する必要があります。設定を解除するパラメータに既定値がなかった場合でも、UnsetSystemParameter() はステータス 1 (成功) を返します。

以下の例では、FullMatchOnly のシステム全体のパラメータ値を設定します。システム全体の既定値が設定されていない場合、プログラムによってこのシステム全体のパラメータが設定されます。システム全体の既定値が設定されている場合は、このシステム全体のパラメータが設定解除された後で設定されます。

ObjectScript

```
#include %IKPublic
SystemwideParameterSet
/* Initial set */
SET stat=##class(%iKnow.Domain).SetSystemParameter($$$IKPFULLMATCHONLY,1)
IF stat=1 {
    WRITE "FullMatchOnly set system-wide to: "
    WRITE ##class(%iKnow.Domain).GetSystemParameter($$$IKPFULLMATCHONLY),!
    QUIT }
ELSE {
    /* Unset and Reset */
    SET stat=##class(%iKnow.Domain).UnsetSystemParameter($$$IKPFULLMATCHONLY)
    IF stat=1 {
        SET stat=##class(%iKnow.Domain).SetSystemParameter($$$IKPFULLMATCHONLY,1)
        IF stat=1 {
            WRITE "FullMatchOnly was unset system-wide",!,"then set to: "
            WRITE ##class(%iKnow.Domain).GetSystemParameter($$$IKPFULLMATCHONLY),!!
            GOTO CleanUpForNextTime }
        ELSE {WRITE "System Parameter set error",stat,!}
    }
    ELSE {WRITE "System Parameter set error",stat,!}
}
CleanUpForNextTime
SET stat=##class(%iKnow.Domain).UnsetSystemParameter($$$IKPFULLMATCHONLY)
IF stat '=1 {WRITE "    Unset error status:",stat}
```

以下の例は、システム全体のパラメータ値を設定するとすぐに、すべてのドメインについてそのパラメータ値が設定されることを示しています。システム全体のパラメータ値を設定した後、個々のドメインについてこの値をオーバーライドできます。

ObjectScript

```
#include %IKPublic
SystemwideParameterUnset
SET stat=##class(%iKnow.Domain).UnsetSystemParameter($$$IKPFULLMATCHONLY)
WRITE "System-wide setting
FullMatchOnly=",##class(%iKnow.Domain).GetSystemParameter($$$IKPFULLMATCHONLY),!!
Domain1Create
SET domn1="mysysdomain1"
SET domo1=##class(%iKnow.Domain).%New(domn1)
DO domo1.%Save()
SET dom1Id=domo1.Id
WRITE "Created the ",domn1," domain ",dom1Id,!
WRITE "FullMatchOnly=",domo1.GetParameter($$$IKPFULLMATCHONLY,.str)," ",str,!!
SystemwideParameterSet
SET stat=##class(%iKnow.Domain).SetSystemParameter($$$IKPFULLMATCHONLY,1)
IF stat=0 {WRITE "SetSystemParameter failed" QUIT}
WRITE "Set system-wide FullMatchOnly=",##class(%iKnow.Domain).GetSystemParameter($$$IKPFULLMATCHONLY),!!
Domain2Create
SET domn2="mysysdomain2"
SET domo2=##class(%iKnow.Domain).%New(domn2)
DO domo2.%Save()
SET dom2Id=domo2.Id
WRITE "Created the ",domn2," domain ",dom2Id,!
WRITE "Domain setting FullMatchOnly=",domo2.GetParameter($$$IKPFULLMATCHONLY,.str)," ",str,!!
DomainParameters
WRITE "New domain ",dom2Id," FullMatchOnly=",domo2.GetParameter($$$IKPFULLMATCHONLY,.str)," ",str,!
WRITE "Existing domain ",dom1Id," FullMatchOnly=",domo1.GetParameter($$$IKPFULLMATCHONLY,.str)," ",str,!!
OverrideForOneDomain
SET stat=domo1.SetParameter($$$IKPFULLMATCHONLY,0)
IF stat=0 {WRITE "SetParameter failed" QUIT}
WRITE "Domain override FullMatchOnly=",domo1.GetParameter($$$IKPFULLMATCHONLY,.str)," ",str,!
CleanupForNextTime
SET stat=##class(%iKnow.Domain).%DeleteId(dom1Id)
SET stat=##class(%iKnow.Domain).%DeleteId(dom2Id)
SET stat=##class(%iKnow.Domain).UnsetSystemParameter($$$IKPFULLMATCHONLY)
```


5.1.4 ドメインへの割り当て

ドメインを作成し、そのドメイン・パラメータを指定 (オプション) すれば、各種コンポーネントをそのドメインに割り当てることができます。

- ・ **ソース・データ** : ドメインの作成後、通常はいくつかの (大概是大量の) テキスト・ソースをドメインにロードすることになります。これにより、InterSystems NLP がインデックスを作成したデータがドメイン内に生成されます。テキスト・ソースのロードは、ほとんどの InterSystems NLP オペレーションで必要になる前提条件です。ファイル、SQL フィールドおよびテキスト文字列を含む、各種のテキスト・ソースがサポートされています。データ型 %String または %Stream.GlobalCharacter (**文字ストリーム・データ**) の SQL フィールドを指定できます。InterSystems NLP によってデータ・ソースのインデックスが作成されれば、以降のオペレーションに影響することなく、元のデータ・ソースを削除できます。データ・ソースの変更は、InterSystems NLP のオペレーションに影響しませんが、そのデータ・ソースをリロードして、インデックス付きのデータをドメインで更新した場合は例外です。
- ・ **フィルタ** : ドメインの作成後、必要に応じて、そのドメインに 1 つまたは複数のフィルタを作成できます。フィルタでは、ロードされたソースのいくつかをクエリから除外するために使用する条件を指定します。したがって、フィルタを使用すると、ドメインにロードしたデータのサブセットに対して InterSystems NLP オペレーションを実行できます。
- ・ **メタデータ** : ドメインの作成後、必要に応じて、ソースをフィルタ処理するための条件として使用できるメタデータ・フィールドを 1 つまたは複数指定できます。メタデータ・フィールドは、インデックスが作成されていないデータであるソースに関連付けたデータです。例えば、テキスト・ソースがロードされた日時は、そのソースのメタデータ・フィールドです。メタデータ・フィールドは、テキスト・ソースをドメインにロードする前に定義する必要があります。
- ・ **skiplist** : ドメインの作成後、必要に応じて、そのドメインに 1 つまたは複数の skiplist を作成できます。skiplist は、クエリから返されることを望まないエンティティ (単語や語句など) のリストです。したがって、skiplist を使用すると、ドメインにロードしたデータ・ソースにある特定のデータ・エンティティを無視する InterSystems NLP オペレーションを実行できます。
- ・ **スマート・マッチング・ディクショナリ** : ドメインの作成後、必要に応じて、そのドメインに 1 つまたは複数のスマート・マッチング・ディクショナリを作成できます。ディクショナリには、インデックス付きのデータとの照合に使用されるエンティティが記述されています。

これらのコンポーネントは、各種の InterSystems NLP クラスおよびメソッドを使用して定義します。InterSystems IRIS **ドメイン・アーキテク**を使用して、メタデータ・フィールドを定義したり、ソースをロードしたり、skiplist やディクショナリを定義したりすることもできます。

メタデータ・フィールドは、ソースをロードする前に定義する必要があります。フィルタ、skiplist およびディクショナリは、いつでも定義または変更することができます。

5.1.5 ドメインからの全データの削除

元のソース・テキストの削除または変更が、そのテキストから InterSystems NLP ドメインにリストおよびロードされたソース・データに影響を及ぼすことはありません。**インデックス作成済みのソース・セットへのソースの追加や削除**は、明示的に行う必要があります。

%DeleteId() 永続メソッドでは、ドメインとそのドメインにリストおよびロードされた全ソース・データを削除します。また、DropData() メソッドを使用すると、ドメイン自体を削除せずに、そのドメインにロードされた全ソース・データを削除できます。いずれのメソッドでもインデックスが作成された全ソース・データが削除され、新しいデータ・ソース・セットを使用して初めからやり直すことが可能になります。

注釈 膨大な数のソースを含んだドメインを削除する場合は、%DeleteId() を使用してドメインを削除する前に、DropData() を使用してデータを削除します。%DeleteId() を使用して、データを内包させたままドメインを削除する場合、InterSystems IRIS はデータ削除を行います。ジャーナリングが無効化されているときにおいても、それぞれのデータ削除についてジャーナリングを実行します。データを削除してからのドメイン削除によって、これらの大規模ジャーナル・ファイルの生成を防ぎます。

IsEmpty() メソッドを使用することで、データがドメインにロードされているか判断できます。

以下の例は、ドメインからのデータの削除を示しています。指定のドメインが存在しない場合は、プログラムによってそのドメインが作成されます。指定のドメインが存在する場合、プログラムがデータの存在をテストします。ドメインにデータが存在する場合は、プログラムによりそのドメインが開かれて、データが削除されます。

ObjectScript

```
DomainCreateOrOpen
SET dname="mytestdomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  IF domoref.IsEmpty() {GOTO RestOfProgram}
  ELSE {GOTO DeleteData }
}
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO RestOfProgram }
DeleteData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!
  GOTO RestOfProgram }
ELSE { WRITE "DropData error",!
  QUIT}
RestOfProgram
WRITE "The ",dname," domain contains no data"
```

5.1.6 全ドメインのリスト

すべてのネームスペースにおける現在のすべてのドメインをリストするには、GetAllDomains クエリを使用できます。詳細は、以下の例を参照してください。

ObjectScript

```
SET stmt=##class(%SQL.Statement).%New()
SET status=stmt.%PrepareClassQuery("%iKnow.Domain","GetAllDomains")
IF status'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(status) QUIT}
SET rset= stmt.%Execute()
WRITE !,"Domains in all namespaces",!
DO rset.%Display()
```

domainId:domainName:namespace:version という形式を使用して、各ドメインが別々の行にリストされます。

Version プロパティは、ドメインの作成時に使用された InterSystems NLP データ構造のバージョンを示す整数になっています。システム・バージョン番号は、InterSystems NLP データ構造の変更がリリースにある場合に変更されます。したがって、InterSystems IRIS の新しいバージョンや新規 InterSystems NLP 機能の導入では、システム・バージョン番号が変わらない場合があります。ドメインの Version プロパティの値が現行の InterSystems NLP システム・バージョンではない場合、ドメインをアップグレードして InterSystems NLP の最新機能を利用することになります。詳細は、“InterSystems NLP の実装”のページの“[NLP データ・アップグレード・ユーティリティ](#)”を参照してください。

既定では、GetAllDomains は、全ネームスペースの現在のドメインをすべてリストします。以下の例に示されているように、%Execute() でブーリアン引数を指定して、現在のネームスペースにドメインのリストを制限できます。

ObjectScript

```
SET stmt=##class(%SQL.Statement).%New()
SET status=stmt.%PrepareClassQuery("%iKnow.Domain","GetAllDomains")
IF status'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(status) QUIT}
SET rset= stmt.%Execute(1)
WRITE !,"Domains in all namespaces",!
DO rset.%Display()
```

ブーリアン値の 1 を指定すると、現在のネームスペースのドメインにリストが制限されます。ブーリアン値の 0 (既定値) を指定すると、全ネームスペースの全ドメインがリストされます。(メモ：リストされる Version プロパティ値は現在のドメイン以外では正しくない場合があります。)

以下を使用すると、現在のネームスペースのすべてのドメインをリスト表示できます。

ObjectScript

```
DO ##class(%SYSTEM.iKnow).ListDomains()
```

このメソッドは、ドメイン ID、ドメイン名、[ソースの数](#)、およびドメインの[バージョン番号](#)をリストします。

5.1.7 ドメイン名の変更

以下の例に示すように、Rename() クラス・メソッドを使用して、現在のネームスペース内の既存ドメインの名前を変更できます。

ObjectScript

```
SET stat=##class(%iKnow.Domain).Rename(oldname,newname)
IF stat=1 {WRITE "renamed ",oldname," to ",newname,!}
ELSE {WRITE "no rename",oldname," is unchanged",!}
```

ドメイン名の変更により、ドメインを開くときに使用する名前が変わり、既存のドメイン ID が新規の名前に割り当てられます。Rename() では、ドメインの現在のインスタンス名は変更されません。名前を変更するには、古いドメイン名が存在して、新しいドメイン名は存在していないことが必要です。

5.1.8 ドメインのコピー

既存のドメインを現在のネームスペースの新規ドメインにコピーするには、[%iKnow.Utils.CopyUtils](#) クラスの CopyDomain() メソッドを使用できます。CopyDomain() メソッドは、ドメインの定義を新規ドメインにコピーして、一意のドメイン名およびドメイン ID を割り当てます。既存のドメインの変更はありません。新規のドメインが存在しない場合、このメソッドにより新規ドメインが作成されます。既定では、このメソッドは[ドメイン・パラメータ設定](#)および[割り当てられたドメイン・コンポーネント](#) (存在する場合) を、既存のドメインからコピーのドメインにコピーします。

また、既定では、CopyDomain() メソッドは[ソース・データ](#)を既存のドメインからコピーのドメインにコピーします。ただし、ソース・データのコピーを要求しても、ソース・データが既存のドメインに存在しない場合、CopyDomain() の処理は失敗します。

以下の例では、“mydomain” という名のドメインおよびそのパラメータ設定とソース・データを “mydupdomain” という名の新規ドメインにコピーします。“mydomain” にソース・データが含まれていないため、(ソース・データをコピーするかどうかを指定する) 3 番目の引数は 0 に設定されています。

ObjectScript

```
DomainMustExistToBeCopied
SET olddom="mydomain"_$PIECE($H,"",2)
SET domo=##class(%iKnow.Domain).%New(olddom)
DO domo.%Save()
IF (##class(%iKnow.Domain).NameIndexExists(olddom))
{WRITE "Old domain exists, proceed with copy",!!}
ELSE {WRITE "Old domain does not exist" QUIT}
CopyDomain
SET newdom="mydupdomain"
IF (##class(%iKnow.Domain).NameIndexExists(newdom))
{WRITE "Domain copy overwriting domain ",newdom,!}
ELSE {WRITE "Domain copy creating domain ",newdom,!}
SET stat=##class(%iKnow.Utils.CopyUtils).CopyDomain(olddom,newdom,0)
IF stat=1 {WRITE !!,"Copied ",olddom," to ",newdom," copying all assignments",!!}
ELSE {WRITE "Domain copy failed with status ",stat,!}
Cleanup
SET stat=##class(%iKnow.Domain).%DeleteId(domo.Id)
WRITE "Deleted the old domain"
```

CopyDomain() メソッドにより、既存ドメインのドメイン設定、ソース・データ、および割り当てられたコンポーネントをすべて新規ドメインにすばやくコピーできます。このメソッドでは、割り当てられたドメイン・コンポーネントすべてをコピーするかまったくコピーしないかに対して、ブーリアン・オプションが用意されています。`%iKnow.Utills.CopyUtils` クラスの他のメソッドでは、割り当てられたコンポーネントのどれを既存ドメイン間でコピーするか指定において、より高度な制御が可能です。

5.2 InterSystems NLP 構成

InterSystems NLP 構成は、ソース・ドキュメントを扱う際の動作を指定します。これは、ソース・データのロード処理時のみ使用されます。構成はそのネームスペースに固有で、1 つのネームスペース内に複数の構成を作成できます。InterSystems NLP はネームスペースの各構成に、一意の整数である構成 ID を割り当てます。構成 ID 値は再利用されません。同じ構成を異なるドメインやソース・テキストのロードに適用できます。InterSystems NLP 構成の定義や使用はオプションで、構成を指定しないと、プロパティの既定値が使用されます。

InterSystems NLP 構成は、以下の 2 つの方法で定義できます。

- ・ `%iKnow.Configuration` クラス・メソッドおよびプロパティを使用する方法 (この章で説明)。
- ・ [ドメイン・アーキテクト](#)を使用して、サポートする言語をドメイン定義の一部として指定する方法。

5.2.1 構成の定義

構成を定義するには、`%iKnow.Configuration` クラスの `%New()` 永続メソッドを使用できます。

`Exists()` メソッドを呼び出すことで、目的の名前の InterSystems NLP 構成が既に存在しているかどうかを判断できます。構成が存在する場合、以下の例に示すように、`Open()` メソッドを使用して、その構成を開くことができます。

ObjectScript

```
IF ##class(%iKnow.Configuration).Exists("EnFr") {
    SET cfg=##class(%iKnow.Configuration).Open("EnFr") }
ELSE { SET cfg=##class(%iKnow.Configuration).%New("EnFr",1,$LB("en","fr"))
    DO cfg.%Save() }
```

5.2.2 構成プロパティの設定

構成では以下のプロパティを定義します。

- ・ **Name** : 構成名には、任意の有効な文字列を指定できます。構成名では、大文字と小文字は区別されません。この構成に割り当てる名前は、現在のネームスペースで一意のものでなければなりません。
- ・ **DetectLanguage** : `Languages` プロパティに複数の言語が指定されている場合に [自動言語識別](#)を使用するかどうかを指定するブーリアン値。このオプションはパフォーマンスに大きな影響を与える可能性があるため、必要な場合以外は設定しないでください。既定値は 0 です (自動言語識別を使用しません)。
- ・ **Languages** : ソース・ドキュメントに含まれる言語、つまり、テストする言語と適用する言語モデルを指定します。使用可能なオプションは、Czech/cs (チェコ語)、Dutch/nl (オランダ語)、English/en (英語)、French/fr (フランス語)、German/de (ドイツ語)、Japanese/ja (日本語)、Portuguese/pt (ポルトガル語)、Russian/ru (ロシア語)、Spanish/es (スペイン語)、Swedish/sv (スウェーデン語)、および Ukrainian/uk (ウクライナ語) です。既定値は、English (en) です。言語は、常に、その言語の ISO 639-1 の 2 文字の省略形式を使用して指定します。このプロパティ値は、(`$LISTBUILD` を使用して) 文字列の InterSystems IRIS リストとして指定します。
- ・ **ユーザ・ディクショナリ** : 定義された [ユーザ・ディクショナリ・オブジェクト](#) の名前、または定義されたユーザ・ディクショナリ・ファイルのファイル・パスのいずれかです。ユーザ・ディクショナリには、InterSystems NLP によるロード・オペ

レーションの際にソース・テキスト・エンティティに適用されるユーザ定義の置換語ペアが記述されています。このプロパティはオプションで、既定値は NULL 文字列です。

- Summarize : ソース・テキストのロード時に要約情報を格納するかどうかを指定するブーリアン値。1 に設定すると、InterSystems NLP が要求するソース情報が生成されて、ロードされたソース・テキストの要約が生成されます。0 に設定すると、この構成オブジェクトで処理されたソースについては、要約を生成できません。通常、このオプションは 1 に設定することをお勧めします。既定値は 1 です。

すべての構成プロパティ (Name は除く) は既定値を割り当てられます。プロパティのディスパッチを使用することで、構成プロパティを取得または設定することができます。

ObjectScript

```
IF cfgOref.DetectLanguage=0 {
    SET cfgOref.DetectLanguage=1
    DO cfgOref.%Save() }
```

プロパティ・ディスパッチを使用してプロパティを変更する前に、最初に新規作成した構成に対して %Save() を実行する必要があります。さらにプロパティ値の変更後、構成に対して %Save() を実行する必要があります。

以下の例では、英語とフランス語を自動言語識別によりサポートする構成を作成しています。その後、英語とスペイン語をサポートする構成に変更しています。

ObjectScript

```
OpenOrCreateConfiguration
SET myconfig="Bilingual"
IF ##class(%iKnow.Configuration).Exists(myconfig) {
    SET cfg=##class(%iKnow.Configuration).Open(myconfig)
    WRITE "Opened existing configuration ",myconfig,! }
ELSE { SET cfg=##class(%iKnow.Configuration).%New(myconfig,1,$LB("en","fr"))
    DO cfg.%Save()
    WRITE "Created new configuration ",myconfig,! }
GetLanguages
WRITE "that supports ", $LISTTOSTRING(cfg.Languages),!
SetConfigParameters
SET cfg.Languages=$LISTBUILD("en","sp")
DO cfg.%Save()
WRITE "changed ",myconfig," to support ", $LISTTOSTRING(cfg.Languages),!
CleanUpForNextTime
SET rnd=$RANDOM(2)
IF rnd {
    SET stat=##class(%iKnow.Configuration).%DeleteId(cfg.Id)
    IF stat {WRITE "Deleted the ",myconfig," configuration" }
}
ELSE {WRITE "No delete this time",! }
```

複数言語と自動言語識別の使用方法の詳細は、“[言語の識別](#)”のページを参照してください。

5.2.3 構成の使用

以下のいずれかの方法で定義された構成を適用できます。

- [DefaultConfig](#) ドメイン・パラメータの定義
- 構成を Init() インスタンス・メソッドの 1 番目の引数に指定して、リスタ・インスタンスを初期化し、構成の既定値をオーバーライドします。
- %iKnow.Source.Listener.SetConfig() の起動
- 構成を loader.ProcessBuffer() または loader.ProcessVirtualBuffer() メソッドの引数として指定します。

5.2.4 全構成のリスト

GetAllConfigurations クエリを使用して、現在のネームスペース内の定義された全構成をリストできます。詳細は、以下の例を参照してください。

ObjectScript

```
SET stmt=##class(%SQL.Statement).%New()
SET status=stmt.%PrepareClassQuery("%iKnow.Configuration","GetAllConfigurations")
IF status'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(status) QUIT}
SET rset= stmt.%Execute()
WRITE "The current namespace is: ", $NAMESPACE,!
WRITE "It contains the following configurations: ",!
DO rset.%Display()
```

1 行に 1 つの構成がリストされ、構成 ID の後に構成パラメータ値が示されます。リストされる値はコロンで区切られます。サポートされる言語のリストを使用して構成が定義されている場合は、GetAllConfigurations はこれらの言語の省略形をコンマで区切って表示します。

以下を使用すると、現在のネームスペースのすべての構成をリスト表示できます。

ObjectScript

```
DO ##class(%SYSTEM.iKnow).ListConfigurations()
```

5.2.5 文字列を正規化する構成の使用法

定義された InterSystems NLP 構成を使用すると、Normalize() メソッドにより、文字列に対してテキストの正規化を実行できます。このメソッドは、以下の例に示すように、文字列の正規化とユーザ・ディクショナリの適用 (任意) の両方を行います。

ObjectScript

```
DefineUserDictionary
SET time=$PIECE($H,"",2)
SET udnname="Abbrev"_time
SET udict=##class(%iKnow.UserDictionary).%New(udnname)
DO udict.%Save()
DO udict.AddEntry("Dr.", "Doctor")
DO udict.AddEntry("Mr.", "Mister")
DO udict.AddEntry("& ", "and")
DisplayUserDictionary
DO udict.GetEntries(.dictlist)
SET i=1
WHILE $DATA(dictlist(i)) {
    WRITE $LISTTOSTRING(dictlist(i),"",1),!
    SET i=i+1
}
WRITE "End of UserDictionary",!!
DefineConfiguration
SET cfg=##class(%iKnow.Configuration).%New("EnUDict"_time,0,$LB("en"),udnname)
DO cfg.%Save()
NormalizeAString
SET mystring="...The Strange Case of Dr. Jekyll & Mr. Hyde"
SET normstring=cfg.Normalize(mystring)
WRITE normstring
CleanUp
DO ##class(%iKnow.UserDictionary).%DeleteId(udict.Id)
DO ##class(%iKnow.Configuration).%DeleteId(cfg.Id)
```

構成に依存しない文字列に対して InterSystems NLP テキスト正規化を実行するには、NormalizeWithParams() メソッドを使用できます。

これらのメソッドは、以下の順序で処理を実行します。

1. 指定されたユーザ・ディクショナリがあれば、それを適用
2. InterSystems NLP 言語モデル処理を実行

3. すべてのテキストを小文字に変換
4. 複数の空白スペース文字を単一の空白文字で置換

5.3 InterSystems NLP ユーザ・ディクショナリ

ユーザ・ディクショナリを使用すると、InterSystems NLP エンジンの既定の動作を拡張できます。ユーザ・ディクショナリは一連の定義ペアで構成され、各定義ペアは文字列を以下の対応するものと関連付けます。

- ・ **意味的属性ラベル** (UDNegation や UDPositiveSentiment) など。この属性に対して文字列は属性マーカとして機能します。意味的属性ラベルを割り当てることで、例えば "tremendous" という単語を、**肯定的な感情**を表す用語として指定できます。
- ・ 文字列の出現ごとに割り当てられるエンティティ・ラベル (UDConcept や UDRelation など)。エンティティ・ラベルを割り当てることで、例えば業界外の人には馴染みのないコンセプトを認識してそのインデックスを作成することを InterSystems NLP に指示できます。
- ・ 文区切りトークン: \end (本来は文区切りしない箇所 で文区切りすることをエンジンに指示)、または \noend (本来は文区切りする箇所 で文区切りを抑制することをエンジンに指示)。
- ・ 使用されている特定の文字列を置き換える置換文字列。置換語のペアを使用して、例えば、使用されているある省略形をすべて、その省略形が表すエンティティに置き換えることができます。

ユーザ・ディクショナリで意味的属性のマーカとしてカスタム用語を定義すると、InterSystems NLP によってその用語の各使用箇所が検出され、ディクショナリの中でその用語に続く属性ラベルに対応する属性によるフラグが、その用語 (およびその用語が使用されている文の一部) に設定されます。**確実性属性**のマーカとして用語を指定する場合は、その用語が使用されている各語句のメタデータとして、確実性レベル c も割り当てる必要があります。

InterSystems NLP の他のすべてのコンポーネントとは異なり、ユーザ・ディクショナリでは、リストしてロードする前のソース・コンテンツが変更されます。つまり、ユーザ・ディクショナリに置換語のペアがあると、以降のオペレーションでは、置換された用語のみが参照されます。例えば、ユーザ・ディクショナリで省略形 "Dr." が "Doctor" に置き換えられていると、InterSystems NLP によってインデックスが作成されたデータに使用されているすべての "Dr." は単語 "Doctor" に置き換えられます。

注釈 ユーザ・ディクショナリの置換でソース・テキストの入力ファイルが変更されることはありませんが、InterSystems NLP の環境で表現されるすべてのソース・テキストは変更され、元に戻すことはできません。元のコンテンツが分析目的で保存されることはないので、元に戻すには、環境を再構築してソースを再ロードする必要があります。このことから、語の置き換えの目的でユーザ・ディクショナリを使用することは通常お勧めできません。

置換語のペアは、NLP テキスト正規化の前に適用されます。ここで、NLP の内部テキスト表現は小文字に変換されます。これにより、置換語のペアでは大文字と小文字が区別されます。したがって、"physician" のインスタンスすべてを "doctor" で置き換えるには、"physician", "doctor", "Physician", "Doctor", そして "PHYSICIAN", "DOCTOR" といった置換語のペアが必要になります。

ユーザ・ディクショナリの定義は任意です。ユーザ・ディクショナリはすべての固有の構成やドメインから独立しています。定義されたユーザ・ディクショナリは**構成プロパティ**として割り当てることができます。1 つの構成に対して割り当て可能なユーザ・ディクショナリは 1 つのみです。ただし、同じユーザ・ディクショナリを複数の構成に割り当てることができます。

また、定義されたユーザ・ディクショナリを、あらゆる構成から独立して、NormalizeWithParams() メソッドに対して指定することもできます。

注釈 ユーザ・ディクショナリはソースがリストされるときにソースに適用されます。ユーザ・ディクショナリを変更しても、既にインデックスが作成されているソースは影響を受けません。

5.3.1 ドメイン・アーキテクトでのユーザ・ディクショナリの定義

対話型のドメイン・アーキテクト・ツールを使用してドメインを作成する際に、[ドメイン設定](#)の過程でユーザ・ディクショナリを定義できます。

5.3.2 オブジェクト・インスタンスとしてのユーザ・ディクショナリの定義

最初にユーザ・ディクショナリ・オブジェクトを作成してから、そのインスタンスを生成します。

ObjectScript

```
SET udict=##class(%iKnow.UserDictionary).%New("MyUserDict")
DO udict.%Save()
DO udict.AddEntry("Dr.", "Doctor")
DO udict.AddEntry("physician", "doctor")
DO udict.AddEntry("Physician", "Doctor")
```

ユーザ・ディクショナリ・オブジェクトを生成するには、追加する定義ペアに適した %iKnow.UserDictionary クラスのメソッドを使用します。例えば、AddConcept() を使用すると文字列をコンセプトのエンティティとして特定でき、AddSentenceNoEnd() を使用すると、文字列の出現によって文区切りが発生しないことを指定できます。

[感情属性](#)など、ユーザ定義の属性用語を追加するには、以下の例のように、適切なインスタンス・メソッドを使用します。

ObjectScript

```
SET udict=##class(%iKnow.UserDictionary).%New("SentimentUserDict")
DO udict.%Save()
DO udict.AddNegativeSentimentTerm("bad")
DO udict.AddNegativeSentimentTerm("horrible")
DO udict.AddPositiveSentimentTerm("good")
DO udict.AddPositiveSentimentTerm("excellent")
```

AddCertaintyTerm() メソッドを使用して確実性属性を割り当てる場合は、以下の例に示すように 2 番目の引数として確実性レベルの整数を指定します。

```
SET udict=##class(%iKnow.UserDictionary).%New("CertaintyUserDict")
DO udict.%Save()
DO udict.AddCertaintyTerm("absolutely", 9)
DO udict.AddCertaintyTerm("presumably", 0)
```

いずれかの汎用属性ラベルを使用してカスタム属性を割り当てるには、汎用の AddAttribute() メソッドを使用します。このメソッドは、属性ラベルを 2 番目の引数の文字列として受け入れます。例：

```
SET udict=##class(%iKnow.UserDictionary).%New("CustomAttrUserDict")
DO udict.%Save()
DO udict.AddAttribute("patient", "UDGeneric1")
```

大文字と小文字を区別する置換語のペアを追加するには、AddEntry() を AddEntry(oldstring,newstring) の形式で使います。必要に応じて、ユーザ・ディクショナリのエントリを追加する位置を指定できます (position の既定はユーザ・ディクショナリ末尾へのエントリ追加です)。InterSystems NLP はユーザ・ディクショナリの順で置換語のペアを適用するので、position を使用すると追加的な置換ができます。例えば、最初に “PA” を “physician’s assistant” で置換してから、“physician” を “doctor” で置換します。

ユーザ・ディクショナリ・オブジェクトを割り当てるには、%New() 構成メソッドの 4 番目の引数にユーザ・ディクショナリ名を指定します。

ObjectScript

```
SET cfg=##class(%iKnow.Configuration).%New("MyConfig",0,$LISTBUILD("en"),"MyUserDict",1)
DO cfg.%Save()
```

5.3.3 ファイルとしてのユーザ・ディクショナリの定義

ユーザ・ディクショナリ・ファイルを生成してから、そのファイルを構成に割り当てることでユーザ・ディクショナリを作成することもできます。

ユーザ・ディクショナリ・ファイルは [UTF-8 形式のエンコード](#) のテキスト・ファイルとする必要があります。

ユーザ・ディクショナリ・ファイルを生成するには、定義ペアごとに別々の行に記述します。

エンティティ・ラベルまたは属性ラベルの割り当ては、@<markerTerm>,<label> の形式に従っている必要があります。確実性属性の場合、その行には @<markerTerm>,UDCertainty,c=<number> の形式で確実性レベルの割り当ても記述する必要があります。

置換語のペアは、<oldString>,<replacementString> の形式に従う必要があります。文字列の前後に空白がある場合にのみ代入や置き換えが適用されるように指定するには、空白の代わりに \ 文字を使用します。指定した文字列の位置で文区切りを適用することを指定するには、<replacementString>. の代わりに /end を指定します。文区切りを適用しないことを指定するには /noend を指定します。

以下にユーザ・ディクショナリ・ファイルのサンプルを示します。

```
Mr.,Mister
Dr.,Doctor
Fr.,Fr
¥UK,United Kingdom
@outstanding,UDPosSentiment
@absolutely,UDCertainty,c=9
@patient,UDGeneric
```

ユーザ・ディクショナリ・ファイルを割り当てるには、そのファイルのフル・パス名を %New() 構成メソッドの 4 番目の引数として指定します。

ObjectScript

```
SET cfg=##class(%iKnow.Configuration).%New(myconfig,0,$LISTBUILD("en"),"C:\temp\udict.txt",1)
DO cfg.%Save()
```


6

プログラムによるテキスト・データのロード

NLP でテキスト・データを分析するには、まず、データ・ソースをドメインにロードする必要があります。これは、以下の 3 通りの方法で実行できます。

- ・ **ドメイン・アーキテクト**を使用して、ドメインのソース・テキストのデータ位置を指定します。**[ビルド]** ボタンで、指定したソースをドメインにロードします。
- ・ **%iKnow.DomainDefinition**すると、ドメイン用のソース・テキストのデータ位置を指定できます。このサブクラスは、このデータをロードするロジックを含んでいる依存クラス内に、%Build() メソッドを生成します。
- ・ ロードとリスタをプログラムで指定して、この章で説明するように、指定したソースをドメインにロードします。

NLP 分析に向けてテキスト・データを準備するために、ドメインではロードとリスタのインスタンスを呼び出す必要があります。ロードはリスタとプロセッサを使用して、テキスト・ソースに対する NLP の処理を監督します。ロードが使用するテキスト・ソースをリスタは識別します。NLP は各種のソース・テキスト・データ用にさまざまなリスタを提供しています。各リスタは既定により、対応するプロセッサを既定のパラメータで自動的に呼び出します。ロードについては、あらゆるタイプのデータ・ソースに対して使用されるものが 1 つ用意されています。

ロードおよびリスタ・オブジェクトは任意の順序で作成できますが、これらは両方ともリスタの AddListToBatch() インスタンス・メソッドとロードの ProcessBatch() インスタンス・メソッド (または他の同等のリスタおよびロード・メソッド) を呼び出す前に作成しておく必要があります。

6.1 ロード

ロード (**%iKnow.Source.Loader**) は、ロード・プロセスを調整するメイン・クラスです。ドメイン用に新しいロード・オブジェクトを作成する必要があります。ロード・オブジェクトを作成する手順は以下のとおりです。

ObjectScript

```
SET domo=##class(%iKnow.Domain).NameIndexOpen("mydomain")
SET domId=domo.Id
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
```

ロードとリスタを作成したら、ソースをリストして処理するためのインスタンス・メソッドを発行します。例えば、バッチ・ロードを実行する場合は、リスタ AddListToBatch() インスタンス・メソッドを発行して、テキスト・ソースをリストします。次に、ロード ProcessBatch() インスタンス・メソッドを発行して、リストされたソースを処理します。AddListToBatch() でマークされた場所をスキャンするリスタをこのロード・メソッドは呼び出してから、これらのドキュメントを読んで NLP エンジンにプッシュするプロセッサを呼び出し、最後に NLP エンジンがロードしたステージング・グローバルを処理する %iKnow.BuildGlobals ルーチン呼び出します。

6.1.1 ロード・エラーのログ

ロード処理が完了したものの、1 つまたは複数のソースのロードにエラーが発生した場合は、そのエラーはエラー・ログに記録されます。さまざまな深刻度のエラーを取得するには、`GetErrors()` メソッド、`GetWarnings()` メソッドおよび `GetFailed()` メソッドを使用できます。例えば、コンテンツのないソース・ファイルのロードを試みると、ロードの失敗というエラー (`GetFailed()`) が発生します。ソース・メタデータ内にエラーがある場合は、ロード・エラーの警告 (`GetWarnings()`) が発生します。

このあらゆる深刻度レベルでのエラー・メッセージのエラー・ログを消去するには、`ClearLogs()` メソッドを使用できます。

6.1.2 ロードの Reset()

ロード処理が予定した方法で完了せず、最初から始める必要がある場合は、以下のように、ローダ・インスタンスの `Reset()` メソッドを起動します。

ObjectScript

```
DO myloader.Reset()
```

6.2 リスタ

テキスト・ファイル、レコード、または NLP によってインデックスを作成したい他の構造化されていないデータのソースをリスタは識別します。つまり、最終的にドメインのソースとなるすべてのテキストを識別します。NLP におけるコンテンツのユニット (単位) はソースで、これは、テキスト・ファイル、SQL テーブルのレコード、RSS ポスト、または他のテキスト・ソースといった分析対象の任意のテキスト・ユニットを表すことができます。

ソースは通常、複数の文を含むテキストです。ただし、ソースは、あらゆるタイプのコンテンツを含むことができます。例えば、123 という数字を含むファイルは、1 つの文を含むソースとして扱われます。コンテンツのないファイルは、ソースとしてリストされません。

すべてのリスタはクラス `%iKnow.Source.Lister` にあり、スキャン可能な独自のタイプのソースを持ちます。例えば、サブクラス `%iKnow.Source.File.Lister` はファイル・システムをスキャンし、サブクラス `%iKnow.Source.RSS.Lister` は、ブログ投稿など、XML ファイル形式の RSS Web フィードをスキャンします。各種のソースに対して、NLP は 7 つのリスタを提供します。また、独自の`カスタム・リスタ`も作成できます。

ほとんどのテキスト・ソースでは、リスタが必要とされます。ただし、文字列として直接指定されるテキストでは、リスタは必要とされません。

`AddListToBatch()` メソッドを使用して、特定のディレクトリ、SQL テーブル、または RSS フィードでソースをチェックするようにリスタに指示することができます。`リスタ・パラメータ`は、実際のリスタ・クラスによって異なります。

6.2.1 リスタの初期化

該当するタイプのリスタの `%New()` メソッドを使用してドメイン ID を指定し、`ドメイン`のリスタ・インスタンスを作成できます。以下の例では、指定のドメイン内に 2 つのリスタを作成しています。

ObjectScript

```
SET domo=##class(%iKnow.Domain).NameIndexOpen("mydomain")
SET domId=domo.Id
SET flister=##class(%iKnow.Source.File.Lister).%New(domId)
WRITE flister,!
SET rlist=##class(%iKnow.Source.RSS.Lister).%New(domId)
WRITE rlist
```


各リスタは、以下のように対応するプロセッサを自動的に呼び出します。

- ・ File.Lister が File.Processor を呼び出します。
- ・ Global.Lister が Global.Processor を呼び出します。
- ・ Domain.Lister が Domain.Processor を呼び出します。
- ・ 他のすべてのリスタが Temp.Processor を呼び出します。%iKnow.Source.Temp.Processor は、ロード・プロセス時に NLP によって自動的に作成されて削除される一時グローバルを処理するため、このような名前を持っています。

各プロセッサは既定のプロセッサ・パラメータを有しており、それらはほとんどの NLP ソースに適合します。したがって、ほとんどの場合、プロセッサまたはプロセッサ・パラメータを指定する必要がありません。プロセッサを指定しない場合、DefaultProcessor() メソッドで示されるように、NLP は既定のプロセッサを使用します。

6.2.2 リスタ・インスタンス既定のオーバーライド

ほとんどの場合、リスタ・インスタンス既定は NLP ソースの処理に適合しています。

構成、プロセッサ、またはコンバータ・オブジェクトのリスタ・インスタンス既定をオーバーライドする場合は、任意でInit() インスタンス・メソッドを使用して、リスタ・インスタンスを初期化できます。Init() を省略すると、既定値が使用されます。

以下にリスタの完全な初期化を示します。

```
Init(config,processor,processorparams,converter,converterparams)
```

これらのどの項目に対しても既定を指定する場合、空の文字列("")を Init() パラメータ値として指定します。

これらのオブジェクトは、SetConfig()、SetProcessor()、および SetConverter() メソッドを使用して別個に初期化することもできます。

- ・ **構成 (Config)**：構成を指定しない場合、NLP は既定の構成を使用します。構成では、テキスト・ドキュメントに含まれる言語のほか、自動言語識別を使用するかどうかを指定します。構成オブジェクトはドメイン固有ではなく、複数のドメインに同じ構成を使用できます。必須ではありませんが、構成は明示的に指定することが推奨されます。
- ・ **プロセッサ**：lister.Init() を使用して、プロセッサとプロセッサ・パラメータを指定できます。プロセッサは NLP にテキストを読み込みます。プロセッサの指定はオプションです。プロセッサを指定しない場合、NLP は既定のプロセッサとその既定のパラメータを使用します。プロセッサを指定する場合は、以下の例に示すように、プロセッサのパラメータ値を指定できます。

ObjectScript

```
SET flister=##class(%iKnow.Source.File.Lister).%New(domId)
SET processor="%iKnow.Source.File.Processor"
SET pparams=$LB("Latin1")
DO flister.Init("",processor,pparams,"","")
```

明示的に指定すると、プロセッサ・サブクラスはリスタ・サブクラスと同じタイプか (例えば、%iKnow.Source.File.Lister は %iKnow.Source.File.Processor を使用します)、またはリスタ・サブクラスに対応するプロセッサ・サブクラスがない場合には、%iKnow.Source.Temp.Processor となるかのいずれかである必要があります。また、独自の**カスタム・プロセッサ**も作成できます。

プロセッサ・パラメータは InterSystems IRIS リストとして指定します。%iKnow.Source.File.Processor では、最初のリスト要素は、使用される文字セットの名前になります (例: "Latin1") %iKnow.Source.Temp.Processor はプロセッサ・パラメータを取りません。

- ・ **コンバータ**：lister.Init() を使用して、ユーザ定義のコンバータとコンバータ・パラメータを指定できます。コンバータはフォーマットされたソース・ドキュメントを平文に変換して、HTMLやXML タグ、PDF フォーマット、または他のテキスト以外のコンテンツを除去します。通常は、それぞれのソース・ドキュメントの形式ごとに別個のコンバータを使用

します。コンバータの指定はオプションです。既定では、コンバータは使用しません。コンバータを使用しない場合は、NLP はフォーマット・コンテンツとテキスト・コンテンツの両方にインデックスを作成します。

6.2.3 リスタによるソースへの ID の割り当て

リスタは次の 2 つの一意の ID をそれぞれのソースに割り当てます。

- ・ ソース ID (内部 ID) : NLP が割り当てる一意の整数値で、NLP の内部処理に使用されます。
- ・ 外部 ID : 一意の識別文字列または数字。外部 ID は、NLP の使用を希望する任意のユーザ指定アプリケーションのためのリンクとして使用されます。外部 ID は以下の構造を持っています。

```
ListerReference:FullReference
```

Lister Reference (リスタ参照) は、このソースのロードに使用されるリスタ・クラスの完全なクラス名、またはリスタ・クラス自身が定義する短いエイリアスになり、接頭語としてコロンが付きます。Full Reference (完全参照) は、リスタ・クラスによって形式が定義される文字列です。これには、グループ名とローカル参照が含まれます。この完全参照からグループ名とローカル参照を派生させ、グループ名とローカル参照から完全参照を再構築するための実装の提供は、リスタ次第です。

例えば、テキスト・ファイルの外部 ID :`FILE:c:\mytextfiles\mydoc.txt` は、以下から構成されます。

- ListerReference : リスタ・クラス・エイリアス :`FILE`
- FullReference : `c:\mytextfiles\mydoc.txt`、これはグループ名 `c:\mytextfiles\` とローカル参照 `mydoc.txt` から構成されます。

SQL テーブルのデータの場合、ListerReference は :`SQL` になります。グループ名は一意の値を含むレコード内のフィールドである `groupfield` で、ローカル参照は行 ID です。

文字列やグローバル変数のデータの場合、ListerReference は :`TEMP` になります。

ここで説明する外部 ID 形式は既定であり、外部 ID 形式は、[SimpleExtIds](#) ドメイン・パラメータを使用して構成できます。

いずれかの ID を使用して、ソースにアクセスできます。`%iKnow.Queries.SourceAPI` クラスには、これらの ID にアクセスするためのメソッドが含まれます。`GetByDomain()` メソッドは、それぞれのソースについて両方の ID を返します。ソース ID を指定すると、`GetExternalId()` メソッドは、外部 ID を返します。外部 ID を指定すると、`GetSourceId()` メソッドは、ソース ID を返します。

リスタ・クラス・エイリアスは、`%iKnow.Source.File.Lister` クラスの `GetAlias()` メソッドを使用して判断できます。エイリアスが存在しない場合は、外部 ID には完全なリスタ・クラス名が含まれます。

6.2.4 リスタの既定値の例

以下に、すべての既定値を使用する、最小限のリスタとローダの例を示します。この例では、ドメインを確立してから、そのドメインのリスタおよびローダ・インスタンス・オブジェクトを作成します。`lister.Init()` は呼び出さずに、[構成](#)、[プロセッサ](#)、および[コンバータ](#)の既定値を使用します。その後、ユーザが定義した `.txt` ファイルと `.log` ファイルのディレクトリをリストして、ロードします。

ObjectScript

```
SET domo=##class(%iKnow.Domain).NameIndexOpen("mydomain")
SET domId=domo.Id
SetListerAndLoader
SET mylister=##class(%iKnow.Source.File.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
UseListerAndLoader
SET install=$SYSTEM.Util.DataDirectory()
SET dirpath=install_mgr\Temp\iris\mytextfiles"
SET stat=mylister.AddListToBatch(dirpath,$LB("txt","log"),0,"")
WRITE "The lister status is ", $System.Status.DisplayError(stat),!
SET stat=myloader.ProcessBatch()
WRITE "The loader status is ", $System.Status.DisplayError(stat),!
```

このドキュメントのほとんどの例では、リスタやローダを使用する前に古いデータを削除しています。古いデータの削除は、デモでこれらの例を繰り返し実行できるようにするために行われています。このドキュメントのほとんどの例では、プロセスとプロセッサ・パラメータを指定せずに、既定を採用しています。このドキュメントの多くの例では、構成についてはその既定値を採用するのではなく、値を指定しています。

6.2.5 リスタ・パラメータ

ソースを指定するメソッドを呼び出す場合は、リスタ・パラメータを指定します。AddListToBatch() リスタ・インスタンス・メソッド (大量のバッチ・ソース・ロード向け) および ProcessList() ロード・インスタンス・メソッド (既存のソース・バッチに少数のソースを追加する場合) には、同じリスタ・パラメータを指定します。

以下の 4 つのリスタ・パラメータの累積によって、NLP インデックス作成用にリストするソースが定義されます。

- Path : ソースが置かれる場所で、文字列で指定します。このパラメータは必須です。
- Extensions : リストするソースを識別する 1 つまたは複数のファイル拡張子接尾語。InterSystems IRIS リスト・データ構造として指定し、それぞれの要素は文字列になります (InterSystems IRIS リスト・データ構造の詳細は “[\\$LIST-BUILD](#)” を参照してください)。既定では、リスタは、パス・ディレクトリにあるファイルの中でデータを含むすべてのファイルを、そのファイル拡張子接尾語に関係なく選択します。これには、ファイル拡張子接尾語を持たないファイルや、テキスト以外のファイル拡張子接尾語 (.jpg など) を持つファイルも含まれます。空のファイルは選択されません。ディレクトリは選択されません。拡張子接尾語パラメータを指定すると、リスタはパス・ディレクトリ内でデータを含むファイルのうち、そのファイル拡張子接尾語を持つ (またはファイル拡張子接尾語のない) ファイルのみを選択します。
- Recursive : パスのサブディレクトリでソースを検索するかどうかを指定するブーリアン値。選択すると、複数レベルのサブディレクトリについてソースを検索します。1 = サブディレクトリを含めます。0 = サブディレクトリを含めません。既定値は 0 です。
- Filter : NLP インデックス作成用にリストするソースを制限するために使用するフィルタを指定する文字列。例えば、ユーザが設計したフィルタによって、ファイル名に指定の部分文字列を持つファイルのみにリスタを制限できます。既定では、フィルタは使用しません (ここで言及されている “フィルタ” は、%iKnow.Filters クラスのフィルタとはまったく異なります。後者のフィルタは、NLP クエリに供給されたインデックス作成済みのソースを含めたり除外したりするために使用されます)。

6.2.6 バッチとリスト

NLP は、あらゆるタイプのソースをロードするために、バッチ・ロード (ProcessBatch()) とリスト・ロード (ProcessList()) という 2 つの方法を提供しています。どちらの方法も同じ処理を実行しますが、実行速度が異なります。どちらを使用するかは、主として、ロードするソースの数によって決まります。原則としては、ロードするソースが 10 個以下の場合は ProcessList() を使用し、100 個以上のソースをロードする場合は ProcessBatch() を使用します。11 ~ 99 個のソースをロードする際にどちらを使用するかは、特定のソースの性質によって決まります。

6.3 リストとロードの例

このセクションの例では、ソースをロードするためのさまざまな方法を示します。

- ・ `listner.AddListToBatch()` と `loader.ProcessBatch()` を使用して、多数のソースをバッチ・ロードします。
- ・ `loader.SetLister()` と `loader.ProcessList()` を使用して、少数のソースをロードしたり、既存のバッチ・ロードにソースを追加します。
- ・ `loader.BufferSource()` と `loader.ProcessBuffer()` を使用して、ソースとして文字列をロードします。もちろん、文字列を含むローカルまたはグローバル変数を指定できます。

また、`loader.ProcessVirtualList()` または `loader.ProcessVirtualBuffer()` を使用して、仮想ソースとしてソースをロードすることもできます。詳細は [“仮想ソースのロード”](#) を参照してください。

6.3.1 ファイルのロード

以下の実行可能サンプル・プログラムは、Windows ディレクトリ `dirpath` 内にある拡張子 `.txt` または `.log` を持つソース・ファイルのバッチ・ロードを実行します。

ObjectScript

```

DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE
        { SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          GOTO SetEnvironment }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { GOTO SetEnvironment }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT}
SetEnvironment
    SET domId=domoref.Id
    IF ##class(%iKnow.Configuration).Exists("myconfig") {
        SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
    ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),"",1)
          DO cfg.%Save() }
CreateListerAndLoader
    SET flister=##class(%iKnow.Source.File.Lister).%New(domId)
    DO flister.Init("myconfig","", "", "", "")
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
UseListerAndLoader
    SET install=$SYSTEM.Util.DataDirectory()
    SET dirpath=install_"mgr\Temp\iris\mytextfiles"
    SET stat=flister.AddListToBatch(dirpath,$LB("txt","log"),0,"")
    WRITE "The lister status is ", $System.Status.DisplayError(stat),!
    SET stat=myloader.ProcessBatch()
    WRITE "The loader status is ", $System.Status.DisplayError(stat),!
QueryLoadedSources
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," sources"

```

この例は、多数のファイルのロードに適したバッチ・ロードを実行します。少数のファイルをロードするには、SetLister() および ProcessList() メソッドを使用します。

6.3.2 SQL レコードのロード

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の実行可能サンプル・プログラムは、Cinema.Review テーブルのレコードのバッチ・ロードを実行します。これは、各レコードの ReviewText フィールド値をソース・テキストとしてロードします。データ型 %String または %Stream.GlobalCharacter (文字ストリーム・データ) のソース・テキスト・フィールドを指定できます。SQL クエリにエラーがある場合、ローダはエラー・ステータスを返します。

SQL データをロードする NLP プログラムは、%iKnow.Source.SQL.Lister を使用する必要があります。このリストでは、常に %iKnow.Source.Temp.Processor を呼び出していますが、パラメータは使用していません。したがって、ユーザ専用のカスタム・プロセッサを作成していない限り、プロセッサを指定する理由はありません。

ObjectScript

```
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO SetEnvironment }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
  GOTO SetEnvironment }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
  QUIT}
SetEnvironment
SET domId=domoref.Id
IF ##class(%iKnow.Configuration).Exists("myconfig") {
  SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),",",1)
  DO cfg.%Save() }
CreateListerAndLoader
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
DO flister.Init("myconfig")
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull,EventDate FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
QueryLoadedSources
WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," sources loaded"
```

この例は、多数の SQL レコードのロードに適したバッチ・ロードを実行します。少数の SQL レコードをロードするには、SetLister() および ProcessList() メソッドを使用します。

また、%SYSTEM.iKnow ユーティリティ・メソッド IndexTable() も使用できます。

6.3.3 添え字付きグローバルの要素のロード

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の実行可能サンプル・プログラムは、添え字付きグローバルの要素をロードします。ここでは %iKnow.Source.Global.Lister を使用して、ProcessList() メソッドに対するリスト・パラメータとしてグローバル名、最初の添え字 (これを含む)、および最後の添え字 (これを含む) を指定します。この例では、^Aviation.AircraftD グローバルを使用しています。これはスパース配列であるため、1 ~ 50,000 の添え字のごく一部しかデータを含んでいません。

ObjectScript

```

DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
  ELSE
    { SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      GOTO SetEnvironment }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { GOTO SetEnvironment }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
        QUIT}
SetEnvironment
  SET domId=domoref.Id
  IF ##class(%iKnow.Configuration).Exists("myconfig") {
    SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
  ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),"",1)
        DO cfg.%Save() }
ListerAndLoader
  SET mylister=##class(%iKnow.Source.Global.Lister).%New(domId)
  DO mylister.Init("myconfig","", "", "", "", "")
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
  SET stat=myloader.SetLister(mylister)
  IF stat '= 1 { WRITE "SetLister error ", $System.Status.DisplayError(stat)
    QUIT}
  SET gbl=" ^Aviation.AircraftD"
  SET stat=myloader.ProcessList(gbl,1,50000)
  IF stat '= 1 { WRITE "ProcessList error ", $System.Status.DisplayError(stat)
    QUIT }
SourceSentenceQueries
  SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
  WRITE "The domain contains ", numSrcD, " sources", !
  SET numSentD=##class(%iKnow.Queries.SentenceAPI).GetCountByDomain(domId)
  WRITE "These sources contain ", numSentD, " sentences"

```

ProcessList() メソッドは、一度に 1 つの添え字レベルしか指定できません。複数の添え字レベルを繰り返すためには、コードを記述して、目的の添え字レベルでこのメソッドを呼び出す必要があります。例えば、セカンド・レベル添え字の 1 および 2 をロードするには、以下のようなコードを記述します。

ObjectScript

```
FOR i=1:1:90000 {  
    SET gbl="^Aviation.NarrativeS("_i_")"  
    SET stat=myloader.ProcessList(gbl,1,2) }
```

これにより、`Aviation.NarrativeS(85879,1)` および `Aviation.NarrativeS(85879,2)` などのグローバルがロードされます。

6.3.4 文字列のロード

以下の実行可能サンプル・プログラムは、1 つのグローバル (または文字列リテラル) をソース・ファイルとしてロードします。文字列をロードする際はリスタは不要であることに注意してください。構成を指定して、ProcessBuffer() メソッドにて適用することができます。

ObjectScript

```

ConfigurationCreateOrOpen
  IF ##class(%Know.Configuration).Exists("EnFr") {
    SET cfg=##class(%Know.Configuration).Open("EnFr") }
  ELSE { SET cfg=##class(%Know.Configuration).%New("EnFr",1,$LB("en","fr"))
        DO cfg.%Save() }
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%Know.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%Know.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
  ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%Know.Domain).%New(dname)
      DO domoref.%Save()
    }

```



```

        SET domId=domoref.Id
        WRITE "Created the ",dname," domain with domain ID ",domId,!
        GOTO CreateLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
        SET domId=domoref.Id
        GOTO CreateLoader }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT}
CreateLoader
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
UseLoader
SET ^a="I drove at 70mph then sped up to 100mph when the light changed."
DO myloader.BufferSource("ref",^a)
DO myloader.ProcessBuffer("EnFr")
QuerySources
WRITE "number of sources:",##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)

```

BufferSource() メソッドの最初の引数には、一意の外部ソース ID を指定します。以下の例では、それぞれのグローバル添え字について別個のソースを作成します。

ObjectScript

```

SET i=1
WHILE $DATA(^a(i)) {
    DO myloader.BufferSource("ref"_i,^a(i))
    DO myloader.ProcessBuffer()
    SET i=i+1 }
WRITE "end of data"

```

また、%SYSTEM.iKnow ユーティリティ・メソッド IndexString() も使用できます。

6.4 ドメイン・コンテンツの更新

ドメインへのソースの初期ロードを実行した後で、ソースを追加または削除することで、このソース・リストを変更できます。ドメインを更新するには、ソース・テキストのセットでの変更への応答を参照します。これをドメインのアップグレードと混同しないでください。ドメインのアップグレードでは、InterSystems IRIS の重要な新規バージョンをインストールした場合、NLP ソフトウェア内での変更への応答を参照します。

6.4.1 ソースの追加

(AddListToBatch() および ProcessBatch() メソッドを使用して) ドメインへのソースの初期ロードを実行した後で、ソース・リストにファイルを追加したい場合があります。これは SetList() および ProcessList() メソッドを使用して行います。ProcessList() メソッドは、AddListToBatch() メソッドと同じパラメータを使用します。

- 一度に 1 つのソースを追加するには、以下のように指定します。SET
stat=myloader.ProcessList("C:\mytextfiles\newfile.txt")
- ソースのディレクトリを追加するには、以下のように指定します。SET
stat=myloader.ProcessList("C:\mytextfiles\logfiles", \$LB("log"), 0, "")

バッチ・ロードにソースを追加する例を以下に示します。

ObjectScript

```

DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      GOTO SetEnvironment }

```

```

DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO SetEnvironment }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT }
SetEnvironment
SET domId=domoref.Id
IF ##class(%Know.Configuration).Exists("myconfig") {
    SET cfg=##class(%Know.Configuration).Open("myconfig") }
ELSE { SET cfg=##class(%Know.Configuration).%New("myconfig", 0, $LISTBUILD("en"), "", 1)
      DO cfg.%Save() }
ListerAndLoader
SET flister=##class(%Know.Source.File.Lister).%New(domId)
DO flister.Init("myconfig", "", "", "", "")
SET myloader=##class(%Know.Source.Loader).%New(domId)
SET stat=myloader.SetLister(flister)
SourceBatchLoad
SET install=$SYSTEM.Util.DataDirectory()
SET dirpath=install_"mgr\Temp\iris\mytextfiles"
SET stat=flister.AddListToBatch(dirpath, $LB("txt"), 0, "")
SET stat=myloader.ProcessBatch()
IF stat '= 1 { WRITE "Loader error ", $System.Status.DisplayError(stat)
              QUIT }
QueryLoadedSources
WRITE "Source count is ", ##class(%Know.Queries.SourceAPI).GetCountByDomain(domId), !
ExpandListofSources
SET elister=##class(%Know.Source.File.Lister).%New(domId)
DO elister.Init("myconfig")
SET stat=myloader.SetLister(elister)
SET addpath=install_"dev\IRIS"
SET stat=myloader.ProcessList(addpath, $LB("txt"), 1, "")
IF stat '= 1 { WRITE "The ProcessList loader status is ", $System.Status.DisplayError(stat)
              QUIT }
QueryTotalSources
WRITE "Expanded source count is ", ##class(%Know.Queries.SourceAPI).GetCountByDomain(domId), !

```

また、%SYSTEM.iKnow ユーティリティ・メソッド IndexFile() および IndexDirectory() も使用できます。

6.4.2 ソースの削除

DeleteSource() メソッドを使用すると、ドメインにロードされたソースを削除できます。このメソッドは、仮想ソースの削除には使用できません。仮想ソースの削除には、別に DeleteVirtualSource() メソッドが用意されています。いずれのメソッドも %SYSTEM.iKnow クラスにあります。

6.5 仮想ソースのロード

仮想ソースとは、静的ではないソースです。例えば、頻繁に変更されるファイルのために仮想ソースを使用する場合があります。仮想ソースの `srcId` は負の整数になります。仮想ソースの外部 ID は、`ListReference` (リスタ・クラスのエイリアス、通常は `:TEMP`) から開始します。

仮想ソースを追加しても、NLPの統計は更新されません。この理由から、ドメインの統計を改訂する手間を招かずに特定目的のためにソースを一時的に追加したい場合は、仮想ソースの使用が適していることがあります。書き込み中のソースなど、継続的に変更されるソースを追加する場合は、仮想ソースを使用すべきです。仮想ソースIDは負の数であるため、仮想ソースは通常のソースと簡単に区別できます。仮想ソースの削除と通常ソースの削除には、異なるメソッドが使用されます。

仮想ソースをロードするには、`loader.SetLister()` と `loader.ProcessVirtualList()` または `loader.BufferSource()` と `loader.ProcessVirtualBuffer()` を使用します。以下のプログラムは、`ProcessVirtualBuffer()` を使用して仮想ソースをロードしています。

ObjectScript

```
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%Know.Domain).NameIndexExists(dname))
        { WRITE("The ",dname," domain already exists",!
```

```

        SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
        GOTO DeleteOldData }
ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      SET domId=domoref.Id
      WRITE "Created the ",dname," domain with domain ID ",domId,!
      GOTO SetEnvironment }
DeleteOldData /* This DOES NOT delete virtual sources */
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
         SET domId=domoref.Id
         GOTO SetEnvironment }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
SetEnvironment
SET config="VSConfig"
IF ##class(%iKnow.Configuration).Exists(config) {
    SET cfg=##class(%iKnow.Configuration).Open(config) }
ELSE { SET cfg=##class(%iKnow.Configuration).%New(config,1)
      DO cfg.%Save() }
CreateLoader
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
VirtualSource
SET node="", (total,status)=0
FOR { SET node=$ORDER(^VendorData(node),1,data) QUIT:node=""
    SET company=$LIST(data,1) QUIT:company=""
    SET address=$LTS($LIST(data,2))
    SET total=total+1
    SET status=myloader.BufferSource("SourceTest"_total,company)
    SET status=myloader.BufferSource("SourceTest"_total,address)
}
SET status=myloader.ProcessVirtualBuffer(config)

SET vsrclist=myloader.GetSourceIds()
FOR i=1:1:$LL(vsrclist) {
    SET srcid=-$LIST(vsrclist,i)
    WRITE "External Id=",##class(%iKnow.Queries.SourceAPI).GetExternalId(domId,srcid)
    WRITE "    Source Id=",srcid,!
    WRITE "    Sentence Count=",##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,$lb(srcid)),!
}

```

%iKnow.Queries.SourceAPI.GetCountByDomain() メソッドは仮想ソースをカウントしないことに注意してください。仮想ソースがロードされたかどうかは、%iKnow.Queries.SourceAPI.GetExternalId(domId,-1) を呼び出すことで判別できます。ここでの -1 は、ロードされた最初の仮想ソースの srcId です。

既定では、多くの NLP クエリは通常のソースのみを処理して、仮想ソースは無視します。これらのクエリを使用して仮想ソースを処理するには、クエリ・メソッドに **vSrcId パラメータ** 値を指定する必要があります。

6.5.1 仮想ソースの削除

%iKnow.Source.Loader クラスは、仮想ソースを削除するための 2 つのメソッドを提供します。

- DeleteVirtualSource() は、ドメインのインデックスが作成された 1 つの仮想ソースを削除します。ドメイン ID (正の整数) と仮想ソース ID (負の整数) を指定します。これによって、このソース・テキストに生成されたすべての NLP エンティティが削除されます。
- DeleteAllVirtualSources() は、指定のドメインのインデックスが作成されたすべての仮想ソースを削除します。これによって、これらのソース・テキストに生成されたすべての NLP エンティティが削除されます。

6.6 ロードされたソース・データのコピーとインデックス再作成

ソースをドメインに正常にロードした後は、そのソースの一部またはすべてを別のドメインにコピーすることが必要な場合もあります。NLP によってそのロードしたソースがコピーされると、そのソースのインデックスも再作成されます。したがって、そのコピーされたソースには別のソース ID とエンティティ ID が付与され、外部 ID は変更されません。

1 つのドメインから別のドメインにコピーまたは再インデックス付けが必要になる理由としては以下が考えられます。

- ・ ドメインのコピーを作成するため。バックアップ・コピーの作成、またはコピーを作成して特定の日にドメインのスナップショットとして機能させることが必要な場合があります。例えば、RSS フィードのインデックスを作成する場合は、このフィードは時間と共に変化し、後日には元のソース・データにアクセスできなくなる可能性があるため、スナップショットの作成が必要な場合があります。
- ・ 元のソース・セットのサブセットを含むドメインを作成するため。この新しいドメインは、扱うのにより小型で、効率性が高く、より簡単になります。このコピーされたソースのサブセットは、コピーするソース ID のリストで指定するか、またはコピーするソースを制限するフィルタで指定できます。例えば、最新のソースのみで構成されるドメインを作成することで、クエリの際に、各クエリの日付でフィルタする必要がなくなる可能性があります。
- ・ 2 つのドメインからマージしたソース・セットを含むドメインを作成するため、または、1 つのドメインからすでにソースを含むドメインにソースを追加するため。
- ・ ソース・セットを抜本的に変更した後にドメイン内のソースのインデックスを再作成するため。例えば、ドメイン内で複数のソースの追加または削除を頻繁に行う場合は、インデックスの作成がすでに最適化されていない可能性があります。(通常のソースの追加および削除では、インデックスのパフォーマンスは低下しません。)ドメインをコピーすることで、現在コピーしているソースのインデックスを再作成し、新しいドメインのインデックス作成を最適化します。
- ・ NLP 言語モデルにリビジョンを適用するため。NLP のリリース・バージョンには通常、その言語モデルの改善点が含まれています。これらには、新しい言語へのサポートの導入やすでにサポートされている言語への改善点が含まれている可能性があります。ドメイン内のソース・セットをコピーすることで、そのソースのインデックスが再作成され、その結果、コピーされたソースに NLP の最新の言語モデルが適用されます。

1つのドメインから別のドメインにコピーまたは再インデックス付けをするには、`%iKnow.Source.Domain.Lister` クラスを使用します。`%New()` メソッドを使用してこのクラスのリスタ・インスタンスを作成可能にするには、新しいドメインを定義しておく必要があります。ドメインは両方とも同じネームスペースに置く必要があります。

以下の例では、`firstdomain` ドメインを生成し、`firstdomain` のコンテンツを `newdomain` という名前の空のドメインにコピーすると、`newdomain` のコンテンツが自動的に再インデックス付けされます。

ObjectScript

```
EstablishAndPopulateFirstDomain
SET domOref=##class(%iKnow.Domain).%New("firstdomain")
DO domOref.%Save()
SET domId=domOref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull,EventDate FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }

TestQueryFirstDomain
WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," sources in the from domain",!

CreateSecondDomain
SET domOref=##class(%iKnow.Domain).%New("newdomain")
DO domOref.%Save()
SET domNewId=domOref.Id

CopyAndReindexFromFirstDomainToSecondDomain
SET newlister=##class(%iKnow.Source.Domain.Lister).%New(domNewId)
SET newloader=##class(%iKnow.Source.Loader).%New(domNewId)
SET stat=newlister.AddListToBatch(domId)
SET stat=newloader.ProcessBatch()

TestQuerySecondDomain
WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domNewId)," sources in the to domain"

CleanupForNextTime
SET stat=##class(%iKnow.Domain).%DeleteId(domId)
IF stat '= 1 {WRITE "Domain delete error:",stat }
SET stat=##class(%iKnow.Domain).%DeleteId(domNewId)
IF stat '= 1 {WRITE "Domain delete error:",stat }
```

AddListToBatch() メソッドによって 2 番目のリスト・パラメータが取得され、コピーするソースを指定できます。また、ソースのリスト (コンマで区切られたソース ID の整数値のリスト) またはフィルタのいずれかを指定できます。以下の例は、コンマで区切られたソース ID のリストを指定することによってコピーするソースを制限する点を除いては、前述の例と同じです。

ObjectScript

```
EstablishAndPopulateFirstDomain
    SET domOref=##class(%iKnow.Domain).%New("firstdomain")
    DO domOref.%Save()
    SET domId=domOref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
    SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull,EventDate FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
TestQueryFirstDomain
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," sources in the from domain",!
CreateSecondDomain
    SET domOref=##class(%iKnow.Domain).%New("newdomain")
    DO domOref.%Save()
    SET domNewId=domOref.Id
SubsetOfSourcesToCopy
    SET subset="1,3,5,7,9,11,13,15,17,19"
CopyAndReindexFromFirstDomainToSecondDomain
    SET newlister=##class(%iKnow.Source.Domain.Lister).%New(domNewId)
    SET newloader=##class(%iKnow.Source.Loader).%New(domNewId)
    SET stat=newlister.AddListToBatch(domId,subset)
    SET stat=newloader.ProcessBatch()
TestQuerySecondDomain
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domNewId)," sources in the to domain"
CleanUpForNextTime
    SET stat=##class(%iKnow.Domain).%DeleteId(domId)
    IF stat '= 1 {WRITE "Domain delete error:",stat }
    SET stat=##class(%iKnow.Domain).%DeleteId(domNewId)
    IF stat '= 1 {WRITE "Domain delete error:",stat }
```

6.6.1 UserDictionary とコピーされたソース

ソースがリストされると、UserDictionary が適用されます。したがって、初期ロード時のソースに行われた UserDictionary の変更はすべて、コピーされたソースに反映されています。ただし、コピー処理はリスト処理でもあるため、新しい UserDictionary を適用して、そのソースをコピーされたソースと同様に更新することもできます。

例えば、ソースの最初のリスト時に使用された UserDictionary で、“Doctor” の省略形として “Dr.” を代入した場合は、この代入がコピーされたソースに反映されます。その後、UserDictionary を変更して “doctor” に “physician” も代入するとします。UserDictionary へのこの変更は、すでにロードされたソースには何の影響も与えません。ソースをコピーすると、この UserDictionary への変更が適用されます。“Dr.” を “doctor” に代入した回数が 0 回になるのは、その代入が初期ロード時のソースにすでに存在するからであり、“physician” の “doctor” への代入はコピーされたソースで実行済みとなっています。

7

テキストのロード時のパフォーマンスに関する考慮事項

NLP では通常、大量のテキスト・データを処理するため、ソース・テキストのロード時に InterSystems IRIS® Data Platform のパフォーマンスに関する以下の考慮事項に注意する必要があります。

- ・ 多数のソースのバッチ・ロードを開始する前に、データベースのジャーナリングを停止します。バッチ・ロードが完了した場合は、ジャーナリングを必ず再開します。ジャーナリングの停止と再開の詳細は、“データ整合性ガイド” の “[ジャーナリング](#)” の章を参照してください。
- ・ 多数のソース (または少数の非常に大きなソース) のバッチ・ロードを開始する前に、この操作の処理に十分な大きさのサイズにグローバル・バッファ・プールを設定します。NLP のインデックス作成によって多数の一時グローバルが作成されます。グローバル・バッファ・プールのメモリがこの一時グローバルの処理に十分な大きさでない場合は、ディスクに書き込まれます。このディスク入出力操作によって、NLP のパフォーマンスは大きな影響を受ける可能性があります。詳細は、“[メモリと開始設定](#)” を参照してください。
- ・ NLP のインデックス作成には、ソース・テキストの占有容量より大幅に大きなディスク容量が必要です。一時グローバルと永続的グローバルに必要な概算容量の詳細は、このドキュメントの “[実装](#)” の章の “グローバルと必要なディスク容量” のセクションを参照してください。
- ・ ソースに必要な数を超える言語サポートを構成しないでください。[NLP 構成](#)では、実際にソースで使用される言語のみが指定される必要があります。ソースがすべて 1 つの言語である場合は、[自動言語識別](#)を指定しないでください。N-gram が言語に要求されないかぎり、[EnableNgrams](#) を [ドメイン・パラメータ](#)に設定しないでください。

8

NLP クエリ

NLP 意味分析エンジンは多数の InterSystems IRIS® Data Platform クエリ API を提供し、これらはテキスト・エンティティおよびこうしたテキスト・エンティティに関する統計を返すために使用されます。例えば、%iKnow.Queries.CrcAPI.GetTop() メソッドは、指定されたドメインで最も頻繁に出現している CRC を返します。また、%iKnow.Queries.CrcAPI.GetCountBySource() は、指定されたソースに出現する一意の CRC の合計数を返します。このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

8.1 クエリのタイプ

提供されるクエリには、以下の 3 つのタイプがあります。これらは、名前の接頭語によって区別されます。

- ・ API : ObjectScript クエリ
- ・ QAPI : InterSystems SQL クエリ
- ・ WSAPI : SOAP でアクセス可能な Web サービス・クエリ

それぞれのタイプに対して、NLP は以下のクエリを提供します。

- ・ **エンティティ** : 1 つ以上のソース内のすべてのエンティティ、最も頻繁に出現するエンティティ、指定の文字列に類似するエンティティなどを返します。
- ・ **CC** : 概念 - 概念のペアを返します。
- ・ **CRC** : 概念 - 関係 - 概念 (ヘッド - 関係 - テール) のシーケンスを返します。
- ・ **パス** : 文内の概念 - 関係 - 概念シーケンスのつながりを返します。パスには、最低 2 つの CRC (CRCRC) が含まれます。
- ・ **文** : 指定された CRC やエンティティなどを含む文を返します。
- ・ **ソース** : 指定された CRC やエンティティなどを含むソースを返します。

8.2 この章で説明するクエリ

この章では、以下のようなよく使用される多数の NLP クエリについて説明し、その例を示します。

- ・ ドメイン内のソースをカウント
- ・ ソース内の文をカウント
- ・ 指定されたエンティティを含むソースをカウント
- ・ ドメイン内の“上位“(最も顕著な)エンティティをリスト
- ・ 指定されたエンティティを含む CRC をリスト (ドメイン全体)
- ・ 指定された CRC を含むソースをカウント
- ・ CRC マスクを満たす文をリスト (エンティティ値とポジションの一致)
- ・ 指定された文字列に類似したエンティティをリスト (ドメイン全体)
- ・ 指定されたエンティティに関連するエンティティをリスト (ドメイン全体)
- ・ ソース内のパスをカウント
- ・ 指定されたソースに類似したソースをリスト (ドメイン全体)
- ・ ソースを要約 (要約文をリスト)

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

この章のクエリの例では、既定の構成を使用しています。ユーザが記述するクエリには、言語環境を確立するために所定の構成が必要になる場合があります。

8.3 クエリ・メソッド・パラメータ

以下のパラメータは、多数のクエリ・メソッドに共通です。

- ・ domainid : ドメイン ID は[ドメイン](#)を特定する整数です。
- ・ result : クエリが 1 つの結果値ではなく値の配列を返す場合は、結果のセットは[参照渡しされます](#) (.result のようにドット接頭語演算子を使用します)。その後で、[ZWRITE](#) を使用して、未加工の InterSystems IRIS リスト形式で全結果セットを表示できます。または、リストから文字列への変換とループ構造を使用して一度に 1 行ずつ返すこともできます (この章の例を参照)。
- ・ page と pagesize (オプション) : メソッドが何千ものレコードを取得して返すのを防ぐために、クエリ API はページング・メカニズムを使用して、返される結果の数をユーザが制限できるようにしています。このメカニズムは、同じ長さのページに結果を分割します。各ページは、pagesize で指定した長さになります。例えば、最初の 10 個の結果が必要な場合は、page に 1、pagesize に 10 と指定します。次のページの結果が必要な場合は、page に 2、pagesize に 10 と指定します。既定値は page=1、pagesize=10 です。
- ・ setop (オプション) : 複数の選択条件にクエリを適用する場合は、Setop 論理演算子によって、クエリで結果セットの UNION (和集合) と INTERSECT (共通集合) のどちらを返すかを指定します。1 (\$\$\$UNION) は、指定された選択条件のいずれかに一致する結果を返します。2 (\$\$\$INTERSECT) は、指定された選択条件のすべてに一致する結果を返します。既定値は \$\$\$UNION です。
- ・ entitylist : エンティティに一致するものを返すクエリ (例 : GetByEntities(), GetRelated(), GetSimilar()) における、InterSystems IRIS によるエンティティのリスト。entitylist のエンティティは、大文字と小文字を混ぜて指定できます。これらは NLP により、小文字に正規化されてインデックスが作成されたエンティティに対してマッチングされます。
- ・ vSrcId : [仮想ソース](#)のソース ID。負の整数で指定します。指定すると、その仮想ソースのエンティティのみがクエリで処理されます。省略した場合は 0 を指定した場合は、通常のソースのみがクエリの対象となり、仮想ソースは無視されます。既定値は 0 です。

8.4 ソースと文のカウント

ロードされたソースの数をカウントするには、`%iKnow.Queries.SourceAPI` クラスの `GetCountByDomain()` メソッドを使用できます。

ロードされた全ソース内の文をカウントするには、`%iKnow.Queries.SentenceAPI` クラスの `GetCountByDomain()` メソッドを使用できます。1 つのソース内の文をカウントするには、`GetCountBySource()` メソッドを使用できます。

以下の例では、`mytextfiles` ディレクトリ内の `.txt` ファイル (`source1.txt`, `source2.txt` など) からロードしたデータを使用して、これらの文カウント・メソッドを示しています。既定の構成の使用となります。

ObjectScript

```
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO ListerAndLoader }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
  QUIT }
ListerAndLoader
SET domId=domoref.Id
SET mylister=##class(%iKnow.Source.File.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
SET stat=myloader.SetLister(mylister)
SET install=$SYSTEM.Util.DataDirectory()
SET dirpath=install_"mgr\Temp\iknow\mytextfiles"
SET stat=myloader.ProcessList(dirpath,$LB("txt"),0,"")
IF stat '= 1 { WRITE "Loader error ", $System.Status.DisplayError(stat)
  QUIT }
SourceSentenceQueries
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The domain contains ", numSrcD, " sources", !
SET numSentD=##class(%iKnow.Queries.SentenceAPI).GetCountByDomain(domId)
WRITE "These sources contain ", numSentD, " sentences", !!
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result, domId, 1, 20)
SET i=1
WHILE $DATA(result(i)) {
  SET extId = $LISTGET(result(i), 2)
  SET fullref = $PIECE(extId, ":", 3, 4)
  SET fname = $PIECE(fullref, "\", $LENGTH(extId), "\")
  SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId, result(i))
  WRITE fname, " has ", numSentS, " sentences", !
  SET i=i+1 }
```

以下の例では、`Aviation.Event` SQL テーブルのフィールドからロードしたデータを使用して、これらの文カウント・メソッドを示しています。この例においては、10 個のデータ・レコード (上位 10) のサンプルのみをロードしています。

ObjectScript

```
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ", dname, " domain already exists", !
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ", dname, " domain does not exist", !
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ", dname, " domain with domain ID ", domoref.Id, !
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ", dname, " domain", !!
  GOTO ListerAndLoader }
```

```

ELSE      { WRITE "DropData error ", $System.Status.DisplayError(stat)
           QUIT }
ListerAndLoader
  SET domId=domoref.Id
  SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
  SET myquery="SELECT Top 10 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
UseLister
  SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
  IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
  SET stat=myloader.ProcessBatch()
  IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
SourceSentenceQueries
  SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
  WRITE "The domain contains ",numSrcD," sources",!
  SET numSentD=##class(%iKnow.Queries.SentenceQAPI).GetCountByDomain(domId)
  WRITE "These sources contain ",numSentD," sentences",!!
  DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,20)
  SET i=1
  WHILE $DATA(result(i)) {
    SET extId = $LISTGET(result(i),2)
    SET fullref = $PIECE(extId,":",3,4)
    SET fname = $PIECE(fullref,"\\",$LENGTH(extId,"\\"))
    SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
    WRITE fname," has ",numSentS," sentences",!
    SET i=i+1 }

```

NLP が何を文と判断するかについての詳細は、“コンセプトの概要” の章の “[NLP が識別する論理テキスト・ユニット](#)” を参照してください。

8.5 エンティティのカウント

指定のエンティティが 1 か所以上出現するソースの数をカウントするには、`%iKnow.Queries.SourceAPI` クラスの `GetCountByEntities()` メソッドを使用できます。このメソッドでは、ロードされたソース内で検索する 1 つ以上のエンティティについてのリストを指定できます。

ここでは、また NLP 全体を通して、“エンティティ” という概念は、検索関連の用語として馴染んだ概念とは大きく異なることに注意してください。例えば、エンティティ “dog” は、“The quick brown fox jumped over the lazy dog.” という文には出現しません。この文に出現するのは、“lazy dog” というエンティティです。エンティティは概念または関係になります。例えば、エンティティ “is” またはエンティティ “jumped over” を含むソースの数をカウントすることが可能です。ただし、この例や現実のほとんどのケースでは、NLP は関係によって関連付けられた 1 つ以上の概念をマッチングさせます。

以下の例は、こうしたクエリ・カウント・メソッドを示しています。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { WRITE "The ",dname," domain already exists",!
    SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
  ELSE
  { WRITE "The ",dname," domain does not exist",!
    SET domoref=##class(%iKnow.Domain).%New(dname)
    DO domoref.%Save()
    WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
    GOTO ListerAndLoader }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { WRITE "Deleted the data from the ",dname," domain",!!
    GOTO ListerAndLoader }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
        QUIT }
ListerAndLoader

```



```

SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
    WRITE "The domain contains ",numSrcD," sources",!
SingleEntityCounts
    SET ent=$LB("NTSB","National Transportation Safety Board",
        "NTSB investigator-in-charge","NTSB oversight","NTSB's Materials Laboratory",
        "FAA","Federal Aviation Administration","FAA inspector")
    SET entcnt=$LISTLENGTH(ent)
    SET ptr=0
    FOR x=1:1:entcnt {
        SET stat=$LISTNEXT(ent,ptr,val)
        WRITE ##class(%iKnow.Queries.SourceQAPI).GetCountByEntities(domId,val)," contain ",val,!
    }
    WRITE "end of listing"

```

8.6 上位エンティティのリスト

NLP には、ドメインのソース・ドキュメントの“上位”エンティティを返すために使用できる以下 3 つのクエリ・メソッドがあります。

- ・ `GetTop()` は、最も頻繁に出現しているエンティティを出現頻度の回数（既定）による降順にてリストします。また、これを使用すれば、分散により最も頻繁に出現しているエンティティをリストすることもできます。このメソッドでは、各エンティティに対して頻度と分散を指定します。これは、上位エンティティの最も基本的なリストとなります。
- ・ `GetTopTFIDF()` は、TFIDF スコアと類似した頻度基準のメトリックを使用して、上位エンティティをリストします。エンティティの用語頻度 (TF : Term Frequency) を逆文書頻度 (IDF : Inverse Document Frequency) と組み合わせることで、このスコアを計算します。用語頻度では、単一ソース内でのエンティティの出現頻度をカウントします。逆文書頻度は、エンティティの分散（“文書頻度”とも言う）の逆数に基づきます。この IDF 頻度を使用することにより、用語頻度を減らします。したがって、少ないパーセンテージのソースにおいて複数回出現するエンティティは、TFIDF スコアが高くなります。大きいパーセンテージのソースにおいて複数回出現するエンティティは、TFIDF スコアが低くなります。
- ・ `GetTopBM25()` は、Okapi BM25 アルゴリズムと類似した頻度基準のメトリックを使用して、上位エンティティをリストします。これは、文書の長さを考慮しつつ、エンティティの用語頻度を逆文書頻度 (IDF) と組み合わせるものです。

これら 3 つのメソッドはすべて、既定にて上位の概念を返しますが、上位の関係を返すために使用することはできません。これら 3 つのメソッドはすべて、フィルタの適用により、使用するソースの対象範囲を制限することができます。

`GetTop()` メソッドは 3 文字未満のエンティティを無視します。`GetTopTFIDF()` および `GetTopBM25()` メソッドは 1 文字および 2 文字のエンティティを返すことができます。

8.6.1 `GetTop()` : 最も頻繁に出現しているエンティティ

NLP クエリは、ソース・ドキュメントにおいて最も頻繁に出現しているエンティティを、頻度または分散度の高いものから順番に返すことができます。各エンティティは、別個のレコードとして InterSystems IRIS リスト形式で返されます。

エンティティ・レコードの形式は以下のとおりです。

- ・ エンティティ ID : NLP が割り当てる一意の整数。

- ・ エンティティ値：文字列として指定されます。
- ・ 頻度：ソース・ドキュメントにエンティティが出現する回数を表す整数値。
- ・ 分散：エンティティを含むソース・ドキュメントの数を表す整数値。

以下のクエリは、このプログラムがロードしたソースに最も頻繁に出現している（上位）エンティティを返します。既定により、これらは概念エンティティとなります。ここでは page (1) および pagesize (50) パラメータを設定して、何個のエンティティを返すかを指定しています。これは（最大で）上位 50 個のエンティティを返します。これはドメインの既定の sorttype を使用しますが、以下のように頻度による降順となっています。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { WRITE "The ",dname," domain already exists",!
          SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE
        { WRITE "The ",dname," domain does not exist",!
          SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
          GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
              GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
          QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
    WRITE "The domain contains ",numSrcD," sources",!
TopEntitiesQuery
    DO ##class(%iKnow.Queries.EntityAPI).GetTop(.result,dmId,1,50)
    SET i=1
    WHILE $DATA(result(i)) {
        SET outstr = $LISTTOSTRING(result(i),"",1)
        SET entity = $PIECE(outstr,"",2)
        SET freq = $PIECE(outstr,"",3)
        SET spread = $PIECE(outstr,"",4)
        WRITE "[",entity,"] appears ",freq," times in ",spread," sources",!
        SET i=i+1 }
    WRITE "Printed the top ",i-1," entities"
```

以下の GetTop() メソッドは分散による上位エンティティを返します。

ObjectScript

```
DO ##class(%iKnow.Queries.EntityAPI).GetTop(.result,dmId,1,50,,, $$SORTBYSREAD)
```

8.6.2 GetTopTFIDF() および GetTopBM25()

これらの 2 つのメソッドは、算出スコアによる降順にて上位エントリのリストを返します。既定により、これらは概念エンティティとなります。これらは異なるアルゴリズムを使用して、スコアをエンティティに割り当てるので、“上位”エンティティの

ストは大幅に異なる場合があります。例えば、以下のテーブルでは、各種メソッドを使用して分析を行った場合における、Aviation.Event データベースで 4 個のエントリの相対順序を示しています。

	“airplane”	“helicopter”	“flight instructor”	“student pilot”
GetTop()	1 番	12 番	17 番	43 番
GetTopTFIDF()	(リストになし)	1 番	4 番	22 番
GetTopBM25()	(リストになし)	3 番	2 番	1 番

GetTop() にて返される Aviation.Event データベースの上位 5 つのエンティティは、“airplane”、“pilot”、“engine”、“flight”、および “accident” となります。これらのエンティティはすべて、半数を超えるソースにて最低 1 回は出現します。これらは頻繁に出現するエンティティである一方で、特定ソースの内容を決定付けるにはほとんど価値のないものです。半数を超えるソースで出現するエンティティには、負の IDF 値が与えられます。このため、これらのエンティティは GetTopTFIDF() および GetTopBM25() のリストに現れません。

以下の例では、GetTopTFIDF() を使用して、上位 50 個のエンティティをリストします。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { WRITE "The ",dname," domain already exists",!
          SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE
        { WRITE "The ",dname," domain does not exist",!
          SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
          GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
              GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT }
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Listener).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
    WRITE "The domain contains ",numSrcD," sources",!
TopEntitiesQuery
    DO ##class(%iKnow.Queries.EntityAPI).GetTopTFIDF(.result,dmId,1,50)
    SET i=1
    WHILE $DATA(result(i)) {
        SET outstr = $LISTTOSTRING(result(i),"",1)
        SET entity = $PIECE(outstr,"",2)
        SET score = $PIECE(outstr,"",3)
        WRITE "[",entity,"] has a TFIDF score of ",score,!
        SET i=i+1 }
    WRITE "Printed the top ",i-1," entities"
```

以下の例では、GetTopBM25() を使用して、上位 50 個のエンティティをリストします。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
    WRITE "The domain contains ",numSrcD," sources",!
TopEntitiesQuery
    DO ##class(%iKnow.Queries.EntityAPI).GetTopBM25(.result,domId,1,50)
    SET i=1
    WHILE $DATA(result(i)) {
        SET outstr = $LISTTOSTRING(result(i),"",1)
        SET entity = $PIECE(outstr,"",2)
        SET score = $PIECE(outstr,"",3)
        WRITE "["entity,"] has a BM25 score of ",score,!
        SET i=i+1 }
    WRITE "Printed the top ",i-1," entities"

```

8.7 CRC クエリ

CRC ([コンセプト - リレーション - コンセプトのシーケンス](#)) を返す NLP クエリは、以下の形式でこれを返します。

- ・ CRC ID : NLP が割り当てて一意の整数。
- ・ ヘッド・コンセプト : 文字列として指定されます。
- ・ リレーション : 文字列として指定されます。
- ・ テール・コンセプト : 文字列として指定されます。
- ・ 頻度 : ソース・ドキュメントに CRC が出現する回数を表す整数値。
- ・ 分散 : CRC を含むソース・ドキュメントの数を表す整数値。

8.7.1 エンティティを含む CRC のリスト

CRC の一般的な使用方法の 1 つに、エンティティ (通常はコンセプト) を指定して、そのエンティティを含む CRC を返す、というものがあります。この場合、1 つまたは複数のソースにエンティティが出現するさまざまなコンテキストを提供し

ます。NLP はすべてのテキストを小文字に正規化するため、一致させるエンティティは小文字で指定する必要があります。

以下のクエリは、指定されたコンセプト ("left wing"、"right wing"、"wings"、"leading edge" および "trailing edge") を含むすべての CRC を、CRC のヘッド概念またはテール概念として返します。GetByEntities() メソッドの page 引数では、より多くの CRC を返すように、25 が設定されていますが、既定値は 10 となります。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { WRITE "The ",dname," domain already exists",!
    SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
  ELSE
  { WRITE "The ",dname," domain does not exist",!
    SET domoref=##class(%iKnow.Domain).%New(dname)
    DO domoref.%Save()
    WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
    GOTO ListerAndLoader }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { WRITE "Deleted the data from the ",dname," domain",!!
    GOTO ListerAndLoader }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
    QUIT}
ListerAndLoader
  SET domId=domoref.Id
  SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
  SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
UseLister
  SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
  IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
  SET stat=myloader.ProcessBatch()
  IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
CRCQuery
  SET myconcepts=$LB("left wing","right wing","wings","leading edge","trailing edge")
  DO ##class(%iKnow.Queries.CrcAPI).GetByEntities(.result,dmId,myconcepts,1,25)
  SET i=1
  WHILE $DATA(result(i)) {
    SET mycrcs=$LISTTOSTRING(result(i),"",1)
    WRITE "[", $PIECE(mycrcs,"",2,4),"]"
    WRITE " appears ", $PIECE(mycrcs,"",5)," times in "
    WRITE $PIECE(mycrcs,"",6)," sources",!
    SET i=i+1 }
  WRITE !,"End of listing"
```

8.7.2 CRC を含むソースのカウント

以下のサンプル・プログラムは、指定された CRC を含むソースの数を返します。CRC を GetCountByCrcs() メソッドに指定するには、(\$LB を使用して) それぞれの CRC を %List として指定してから、これらの CRC を %List としてグループ化する必要があります。詳細は、以下の例を参照してください。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { WRITE "The ",dname," domain already exists",!
    SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
  ELSE
  { WRITE "The ",dname," domain does not exist",!
    SET domoref=##class(%iKnow.Domain).%New(dname)
    DO domoref.%Save()
    WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  }
```

```

        GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
        GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
CRCCount
SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
SET mycrs=$LB($LB("leading edge","of","wing"),$LB("leading edge","of","right wing"),
              $LB("leading edge","of","left wing"),$LB("leading edges","of","wings"),
              $LB("leading edges","of","both wings"))
SET numSrc=##class(%iKnow.Queries.SourceAPI).GetCountByCrcs(domId,mycrs)
WRITE "From ",numSrcD," indexed sources there are ",!
WRITE numSrc," sources containing one or more of the following CRCs:",!
FOR i=1:1:$LISTLENGTH(mycrs) {
    WRITE $LISTTOSTRING($LIST(mycrs,i)," "),!
}

```

GetCountByCrcs() メソッドは、指定された CRC のいずれかを含むソースの数を返します。

8.7.3 CRC マスクを満たす 1 つまたは複数の文のリスト

CRC マスクを使用して、特定 CRC ポジションのエントリ値を指定することができます。それぞれの CRC は、ヘッド、関係、テールという 3 つのポジションを備えています。CRC マスクを使用して、各ポジションのエントリ値またはワイルドカードを指定できます。CRC マスクにより、1 つ以上のポジション値が一致する CRC を含むソースや文をリストできます。GetByCrcMask() による部分的 CRC 一致ではポジションとエントリ値の両方を指定するため、GetByEntities() より一致条件が制限されますが、GetByCrcs() ほどは制限されません。

以下の例では、ヘッド・ポジションでエントリ“student pilot”に一致する CRC マスクを使用すると同時に、ワイルドカードを使用して、CRC の関係およびテール・ポジションにおいて任意の値を許可しています。GetByCrcMask() メソッドは、各ソースのそれぞれの文に対してこのマスクをマッチングさせ、ヘッド・ポジションに“student pilot”の CRC を含む文の ID とテキストを返します。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
        GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild

```



```

SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
CRCMaskSentencesBySource
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,100)
SET i=1
WHILE $DATA(result(i)) {
SET srcId = $LISTGET(result(i),1)
SET extId = $LISTGET(result(i),2)
SET srcname = $PIECE($PIECE(extId,":",3,4),"\",$LENGTH(extId,"\"))
SET numSents = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
WRITE numSents," sentences in ",srcname,!
SET stat=##class(%iKnow.Queries.SentenceAPI).GetByCrcMask(.sentresult,domId,"student pilot",
$$$WILDCARD,$$$$WILDCARD,srcId)
SET i=i+1
FOR j=1:1:20 {
IF $DATA(sentresult(j)) {
SET sent = $LISTTOSTRING(sentresult(j),",",1)
SET sentId = $PIECE(sent,":",3)
WRITE "The SentenceId is ",sentId," in source ",srcname,":",!
WRITE " ",##class(%iKnow.Queries.SentenceAPI).GetValue(domId,sentId),!
}
ELSE { WRITE "Listed ",j-1," sentence that match the CRC mask",!!
QUIT }
}
}
}

```

8.8 類似エンティティのリスト

指定の文字列に類似した一意のエンティティをリストできます。以下のいずれかに該当する場合、エンティティは類似しています。

- ・ 文字列がエンティティと同じである場合。
- ・ 文字列がエンティティの単語の 1 つである場合。
- ・ 文字列がエンティティの単語の 1 つの最初の文字である場合。

類似性クエリでは、それぞれの一意のエンティティ(ヘッド・コンセプトまたはテール・コンセプト)と共に、その頻度と分散の整数値が、それらの整数値の降順で返されます。類似性クエリでは、リレーションはマッチングされません。NLP では常にそうであるように、マッチングでは大文字と小文字は区別されず、すべてのエンティティが小文字で返されます。類似性クエリでは語幹解析ロジックは使用されず、“cat”と指定すると“cats”と“category”の両方が返されます。

以下の例では、文字列“student pilot”に類似したエンティティがリストされます。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
SET domoref=##class(%iKnow.Domain).%New(dname)
DO domoref.%Save()
WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)

```

```

        QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," total sources",!!
SimilarEntityQuery
    WRITE "Entities similar to 'Student Pilot':",!
    DO ##class(%iKnow.Queries.EntityAPI).GetSimilar(.simresult,dmId,"student pilot",1,50)
    SET j=1
    WHILE $DATA(simresult(j)) {
        SET outstr = $LISTTOSTRING(simresult(j),",",1)
        SET entity = $PIECE(outstr,",",2)
        SET freq = $PIECE(outstr,",",3)
        SET spread = $PIECE(outstr,",",4)
        WRITE "(",entity,") appears ",freq," times in ",spread," sources",!
        SET j=j+1 }

```

エンティティの類似性を左右する既定の**ドメイン・パラメータ**設定は、[EnableNgrams](#) というブーリアン値です。

8.8.1 パーツと N-gram

GetSimilar() および GetSimilarCounts() メソッドは、類似性を検索する場所を指定するモード・パラメータを備えています。利用可能な値は以下の 2 つです。

- ・ **\$\$\$USEPARTS** を使用すると、NLP は各パーツ (単語) の先頭をマッチングさせて類似性をチェックします。英語をはじめとするほとんどの言語のテキストでは、これは一般的に優先設定になります。**\$\$\$USEPARTS** は既定値です。
- ・ **\$\$\$USENGRAMS** を使用すると、NLP は単語と単語内の言語単位 (N-gram) をマッチングさせて類似性をチェックします。このモードは、ソース・テキストの言語が単語を組み合わせる場合に使用されます。例えば、**\$\$\$USENGRAMS** は、頻繁に複合語が形成されるドイツ語でよく使用されます。単語の組み合わせを行わない英語では、**\$\$\$USENGRAMS** は使用されません。**\$\$\$USENGRAMS** は、**EnableNgrams** **ドメイン・パラメータ**のセットを持つドメインでのみ使用できます。

8.9 関連するエンティティのリスト

2 つのエンティティが 1 つの CRC に出現する場合は、これらのエンティティは関連しています。既定では、関連するエンティティはヘッド概念とテール概念のいずれかになります (この既定をオーバーライドするには、“ポジションによる制限” (後述) を参照してください)。

NLP がどのようにして関連するエンティティを返すかを、以下の例で示します。iKnow はまず、何個の CRC にエンティティ “student pilot” が含まれているかを判断して、これらの CRC をリストします (この小さな例では、単純にすべての CRC を読んで “student pilot” に関連するものをチェックできますが、大きなソースのコレクションでは、これは不可能となります)。次に、このサンプル・プログラムは、“student pilot” に関連するすべてのエンティティを、テールまたはヘッドとしてリストします (これらの関係は、以前にリストされた CRC に対してこれらのエンティティをマッチングさせることで確認できます)。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," total sources",!!
ContainCRCQuery
    SET crccount = ##class(%iKnow.Queries.CrcAPI).GetCountByEntities(domId,$LB("student pilot"))
    WRITE crccount," CRCs contain 'student pilot'",!
    DO ##class(%iKnow.Queries.CrcAPI).GetByEntities(.result,domId,$LB("student pilot"),1,crccount)
    SET i=1
    WHILE $DATA(result(i)) {
      WRITE $LISTTOSTRING(result(i),"",1),!
      SET i=i+1 }
    SET relcount = ##class(%iKnow.Queries.EntityAPI).GetRelatedCount(domId,$LB("student pilot"))
    WRITE !,relcount," entities are related to 'student pilot':",!
RelatedEntityQuery
    DO ##class(%iKnow.Queries.EntityAPI).GetRelated(.rresult,domId,$LB("student pilot"),1,relcount)
    SET j=1
    WHILE $DATA(rresult(j)) {
      WRITE $LISTTOSTRING(rresult(j),"",1),!
      SET j=j+1 }

```

8.9.1 ポジションによる制限

エンティティのポジションは、ヘッド・コンセプト、リレーション、またはテール・コンセプトになります。既定では、GetRelated() メソッドは、ポジションにかかわらず関連するすべての概念を返し、関係は返しません。この既定は、8 番目のパラメータ (positiontomatch) にマクロ定数を指定することで変更できます。使用可能な定数は以下のとおりです。

定数	値	意味
\$\$\$USEPOSM	1	ヘッド・コンセプト
\$\$\$USEPOSR	2	リレーション
\$\$\$USEPOSMR	3	ヘッド・コンセプトとリレーション
\$\$\$USEPOSS	4	テール・コンセプト
\$\$\$USEPOSMS (既定)	5	ヘッド・コンセプトとテール・コンセプト
\$\$\$USEPOSRS	6	リレーションとテール・コンセプト
\$\$\$USEPOSALL	7	ヘッド・コンセプト、リレーション、およびテール・コンセプト

以下の例では、関連するヘッド概念と関連するテール概念を分離させます (\$\$\$USEPOSM は、指定の文字列は CRC のヘッド概念で、関連するエンティティはテール概念であることを意味します)。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { WRITE "The ",dname," domain already exists",!
          SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE
        { WRITE "The ",dname," domain does not exist",!
          SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
          GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
             GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
          QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," total sources",!!
ContainCRCQuery
    SET crccount = ##class(%iKnow.Queries.CrcAPI).GetCountByEntities(domId,$LB("student pilot"))
    WRITE crccount," CRCs contain 'student pilot'",!
    DO ##class(%iKnow.Queries.CrcAPI).GetByEntities(.result,domId,$LB("student pilot"),1,crccount)
    SET i=1
    WHILE $DATA(result(i)) {
        WRITE $LISTTOSTRING(result(i),"",1),!
        SET i=i+1 }
    SET relcount = ##class(%iKnow.Queries.EntityAPI).GetRelatedCount(domId,$LB("student pilot"))
    WRITE !,relcount," entities are related to 'student pilot':",!
ListRelatedHeadsQuery
    DO ##class(%iKnow.Queries.EntityAPI).GetRelated(.mresult,domId,$LB("student
pilot"),1,relcount,"",$$$$USEPOSM)
    WRITE !,"The following have 'student pilot' as a head:",!
    SET j=1
    WHILE $DATA(mresult(j)) {
        WRITE $LISTTOSTRING(mresult(j),"",1),!
```

```

        SET j=j+1 }
ListRelatedTailsQuery
DO ##class(%iKnow.Queries.EntityAPI).GetRelated(.sresult,domId,$LB("student
pilot"),1,relcount,"","",$USEPOSS)
WRITE !,"The following have 'student pilot' as a tail:",!
SET k=1
WHILE $DATA(sresult(k)) {
    WRITE $LISTTOSTRING(sresult(k),"",1),!
    SET k=k+1 }

```

8.10 パスのカウント

以下の例は、50 個のソースのパスの数と文の数を示しています。通常は、ソース内の文よりも多くのパスが存在します。ただし、いくつかのソース内のパスよりも文が多く存在することは可能です。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    WRITE ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)," total sources",!!
PathCountBySource
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,50)
    SET i=1
    WHILE $DATA(result(i)) {
        SET srcId = $LISTGET(result(i),1)
        SET extId = $LISTGET(result(i),2)
        SET fullref = $PIECE(extId,":",3,4)
        SET fname = $PIECE(fullref,"\\",$LENGTH(extId,"\\"))
        SET numPathS = ##class(%iKnow.Queries.PathAPI).GetCountBySource(domId,$LB(srcId))
        SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
        WRITE numPathS," paths and ",numSentS," sentences in ",fname,!
        SET i=i+1 }

```

8.11 類似ソースのリスト

NLP 意味分析エンジンは、指定のソースに類似するソースをリストできます。ソース間の類似性は、両方のソースに出現するエンティティ（重複）の数と、重複を含むソース・コンテンツのパーセンテージによって決まります。

GetSimilar() メソッドは指定のソースに対して、ソースの類似性を計算することができます。大量の類似ソースが存在する可能性があるため、通常はこのメソッドをフィルタと共に使用することで、考慮対象のソース・セットを制限します。GetSimilar() は、以下の 2 つのアルゴリズムから選択可能であり、それぞれがアルゴリズム・パラメータを持っています。

- ・ 項目の基本的類似性 (\$\$\$SIMSRCSIMPLE、既定) : 使用できるアルゴリズム・パラメータは、“ent” (エンティティの類似性、既定)、“crc” (コンセプト - リレーション - コンセプトのシーケンス)、または“cc” (コンセプト - コンセプトのペア) となります。
- ・ 意味的優位性の計算 (\$\$\$SIMSRCDOMENTS) : アルゴリズム・パラメータは、指定ソース内での優位性の高いエンティティでもある高優位性エンティティを含むソースに対して、類似性を制限するブーリアン・フラグとなります。

NLP は、類似するそれぞれのソースについて、以下の形式で要素のリストを返します。

srcId,extId,percentageMatched,percentageNew,nbOfEntsInRefSrc,nbOfEntsInCommon,nbOfEntsInSimSrc,score	
要素	説明
srcId	ソース ID。NLP が割り当てる整数です。
extId	ソースの外部 ID (文字列値)。
percentageMatched	一致元ソースと同じソース・コンテンツのパーセンテージ。
percentageNew	新しいソース・コンテンツのパーセンテージ。新しいコンテンツは、一致元ソースと一致しないコンテンツです。
nbOfEntsInRefSrc	参照される (このソースに対してマッチングされる) ソース内の一意のエンティティの数。
nbOfEntsInCommon	両方のソースにある一意のエンティティの数。
nbOfEntsInSimSrc	このソース内の一意のエンティティの数。
スコア	小数で表される類似性スコア。同一のソースは類似性スコアが 1 となります。

以下の例は、類似するソースのリストを示しています。最初に、GetByEntities() を使用して適切なエンティティのリストを選択することにより、テスト・ソースのセットを engine failure incident (エンジン故障事故) を述べる可能性のあるものに制限しています。次に、GetSimilar() を使用して、これらのテスト・ソースと類似するソースを検出します。これは類似事象パターンを示している可能性があります。GetSimilar() は、既定の類似性アルゴリズム (\$\$\$SIMSRCSIMPLE) およびその既定アルゴリズム・パラメータ (“ent”) を使用しています。このプログラムは、高類似性スコア (>.33) 持つ類似ソースのみを表示します。類似性の表示では、ソースの外部 ID は省略されています。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
```



```

SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
    SET totsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
    WRITE totsrc," total sources",!
SimiarSourcesQuery
    SET engineents = $LB("engine","engine failure","engine power","loss of
power","carburetor","crankshaft","piston")
    DO ##class(%iKnow.Queries.SourceAPI).GetByEntities(.result,domId,engineents,1,totsrc)
    SET i=1
    WHILE $DATA(result(i)) {
        SET src = $LISTTOSTRING(result(i),"",1)
        SET srcId = $PIECE(src,"",1)
        WRITE "Source ",srcId," contains an engine incident",!
        DO ##class(%iKnow.Queries.SourceAPI).GetSimilar(.sim,domId,srcId,1,50,"",$SIMSRCSIMPLE,$LB("ent"))

        SET j=1
        WHILE $DATA(sim(j)) {
            SET simlist=$LISTTOSTRING(sim(j))
            IF $PIECE(simlist,"",8) > .33 {
                WRITE "    similar to source ",$PIECE(simlist,"",1),": "
                WRITE $PIECE(simlist,"",3,8),! }
            SET j=j+1 }
        SET i=i+1 }

```

8.12 ソースの要約

NLP 意味分析エンジンは、最も関連性のある文を返すことでソース・テキストを要約できます。これは、ソース・テキストのコンテンツ全体に最も類似する文を選択して、それらの文をユーザが指定した数だけ、元の文の順序で返します。NLP はそれぞれの文について内部関連性スコアを計算して、関連性を判断します。ソース・テキストに何度も出現する概念を含む文は、ソース・テキストに一度しか出現しない概念を含む文よりは、要約に含まれる可能性が高くなります。NLP は、各概念の総合的な頻度、ソースにおいて最も頻度の高い概念に対する各概念の類似性、および他の要因を考慮します。

ソースの要約は、ソースのロード時に構成で Summarize プロパティが 1 に設定された場合にのみ利用可能となります。既定の構成では、Summarize=1 が指定されます。

したがって、要約の正確さは、以下の 2 つの要因によって左右されます。

- ・ ソース・テキストは、意味のある頻度分析が可能な十分な大きさであることが必要ですが、大き過ぎることも禁物です。NLP の要約は、長い章や項目のテキストで最も効果を発揮します。本一冊分のテキストは、章ごとに要約する必要があります。
- ・ ユーザが読んで理解できる要約テキストが返される文によって構成されるように、要約内の文の数は元のテキストの十分大きなサブセットであることが必要です。要約割合の最小値は、25 % ~ 33 % となり、テキストの内容により異なります。

NLP には、以下の 3 つの要約メソッドが用意されています。

- ・ GetSummary() では、要約テキストの各文を個々の結果として返します。この文の ID は、返された各文の先頭要素として返されます。
- ・ GetSummaryDirect() では、要約テキストを 1 つの文字列として返します。既定では、この文字列内の文は省略記号 (空白スペース、3 つのピリオド、空白スペース) によって区切られます。例えば、“This is sentence one. ... This is sentence two.” のようになります。必要に応じて、別の文区切りを指定できます。このメソッドでは、複数の文を 1 つ

の文字列に連結するため、InterSystems IRIS の最大文字列長より長い文字列の作成を試みる場合があります。最大文字列長に達すると、InterSystems IRIS では、このメソッドの isTruncated というブーリアン出力パラメータが 1 に設定され、残りのテキストが切り捨てられます。

- ・ GetSummaryForText() では、特定のソース ID を指定するのではなく、ユーザ指定文字列の要約を直接編集する方式をサポートしています。

NLP が何を文と判断するかについての詳細は、“コンセプトの概要” の章の “NLP が識別する論理テキスト・ユニット” を参照してください。

以下の例では、101 以上の文を含むものを見つけるまで、ドメイン内のソース・テキストをチェックします。次に、GetSummary() を使用して、そのソースを元の文の半分に要約します。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
  GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
  QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceSentenceTotals
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The domain contains ",numSrcD," sources",!
SET numSentD=##class(%iKnow.Queries.SentenceAPI).GetCountByDomain(domId)
WRITE "These sources contain ",numSentD," sentences",!!
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domainId)
SentenceCounts
FOR i=1:1:numSrcD {
  SET srcId = $LISTGET(result(i),1)
  SET extId = $LISTGET(result(i),2)
  SET fullref = $PIECE(extId,":",3,4)
  SET fname = $PIECE(fullref,"\",$LENGTH(extId,"\"))
  SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
  IF numSentS > 100 {WRITE fname," has ",numSentS," sentences",!
    GOTO SummarizeASource }
}
QUIT
SummarizeASource
SET sumlen=$NUMBER(numSentS/2,0)
WRITE "total sentences=",numSentS," summary=",sumlen," sentences",!!
DO ##class(%iKnow.Queries.SourceAPI).GetSummary(.sumresult,domainId,srcId,sumlen)
FOR j=1:1:sumlen { WRITE "[S",j,": ",$LISTGET(sumresult(j),2),! }
WRITE !,"END OF ",fname," SUMMARY",!!
QUIT
```

\$NUMBER は、要約文の指定数を確実に整数にするために使用されています。\$LISTGET は、文 ID を削除して、文テキストだけを返すために使用されています。

以下の例では、GetSummaryDirect() を使用して、連結された 1 つの文字列として同じ要約が返されます。次に、[\\$EXTRACT](#) を使用して、その文字列を以下のように表示するために、38 文字の行に分割します。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceSentenceTotals
    SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
    WRITE "The domain contains ",numSrcD," sources",!
    SET numSentD=##class(%iKnow.Queries.SentenceAPI).GetCountByDomain(domId)
    WRITE "These sources contain ",numSentD," sentences",!!
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId)
SentenceCounts
    FOR i=1:1:numSrcD {
        SET srcId = $LISTGET(result(i),1)
        SET extId = $LISTGET(result(i),2)
        SET fullref = $PIECE(extId,"",3,4)
        SET fname = $PIECE(fullref,"\\",$LENGTH(extId,"\\"))
        SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
        IF numSentS > 100 {WRITE fname," has ",numSentS," sentences",!
          GOTO SummarizeASource }
    }
    QUIT
SummarizeASource
    SET sumlen=$NUMBER(numSentS/2,0)
    WRITE "total sentences=",numSentS," summary=",sumlen," sentences",!!
    SET summary = ##class(%iKnow.Queries.SourceAPI).GetSummaryDirect(domId,srcId,sumlen)
FormatSummaryDisplay
    SET x=1
    SET totlines=$LENGTH(summary)/38
    FOR i=1:1:totlines {
        WRITE $EXTRACT(summary,x,x+38),!
        SET x=x+39 }
    WRITE !,"END OF ",fname," SUMMARY"
```

8.12.1 カスタム・サマリ

NLP では、summaryConfig パラメータ文字列を指定することで、ソースのカスタム・サマリを生成できます。カスタム・サマリは、NLP で生成されたサマリの内容を自身のニーズに合わせて調整したいユーザーのために提供されています。カスタム・サマリにより、文の要約への絶対的な包含、優先的な包含、または文の要約からの絶対的な除外が可能になります。例えば、タイトル、署名、著作権、抜粋、または要旨など常に同じ位置に現れるソースの標準コンポーネントを包含または除外することができます。また、指定した単語を含む文を、絶対的または優先的に包含/除外することもできます。

ソースの要約処理では、始めに各文に数値的な要約重要度が付与された後、最も重要度の高い文の適切な番号を選択することにより、要約を作成します。要約メソッドに対して `summaryConfig` パラメータを指定することで、このランキングを左右することもできます。

`summaryConfig` パラメータの値は、1 つ以上の仕様から成る文字列となります。各仕様は垂直バー (|) で仕切られた 3 つの要素から成ります。例えば、"`s|2|false`" となります。垂直バー (|) を使用して、複数の仕様を連結できます。例えば、"`s|1|true|s|2|false`" となります。`summaryConfig` パラメータの既定値は空の文字列です。

以下に従って、要約を構成することで、文を選択することができます。

- ・ 常に文を包めます (または一切包めない)。
- 文の番号によるもの: "`s|1|true`" は、文 (s) の番号 1 を常に要約に包める (true) という意味です。例えば、ソースが常にタイトルを含んでいる場合、要約に最初の文 (タイトル) を常に含むことが効果的となります。各ソースの 2 行目が署名である場合、かつこれを要約に含めない場合は、これを "`s|2|false`" として指定できます。また、次のように最後の文を要約に含めないことを指定できます。"`s|-1|false`" は、すべてのソースが写しの参照や刊行物の引用で終わっている場合に適しています。ソースの文は 1 から順番に番号を割り振るか、もしくは、ソースの最後から -1、-2 などと番号を割り振っていきます。
- 単語によるもの: "`w|requirement|true`" は、単語 (w) `requirement` を含む文はいずれも、常に要約に含める (true) という意味です。また、特定の単語を含む文を除外することもできます。例えば、"`w|foreign|false`" により、単語 `foreign` を含む文はすべて要約から除外されます。“単語” を 1 つ以上の単語でまとめることもできます。その場合、空白で区切られた複数の単語の文字列で構成されます。部分語の文字列で構成することはできません。単語は正規化されるので、すべて小文字で指定する必要があります。
- ・ 文に高い要約重要度を付与します。これにより、文が要約内で出現する可能性が高くなります。使用できる重要度の値は、0 ~ 9 の整数となります。
- 文の番号によるもの: "`s|1|3`" は、文 (s) の番号 1 が 3 の要素により増加する要約重要度を備えているという意味です。例えば、内容をある程度記述している場合は、ソースのタイトル (最初の文) を含める必要がありますが、直接的な記述でない場合 (文学的な引用など) は、その必要はありません。ソースの文は 1 から順番に番号を割り振るか、もしくは、ソースの最後から -1、-2 などと番号を割り振っていきます。
- 単語によるもの: "`w|requirement|2`" は、単語 (w) `requirement` を含む文はいずれも、2 の要素により増加する要約重要度を備えているという意味です。“単語” を 1 つ以上の単語でまとめることもできます。その場合、空白で区切られた複数の単語の文字列で構成されます。部分語の文字列で構成することはできません。単語は正規化されるので、すべて小文字で指定する必要があります。

連結演算により、複数の要約のカスタマイズを指定することができます。以下はその例で

す。"`s|1|true|s|2|false|w|surgery|3|w|hypnosis|false`" は、常に最初の文を含め、2 番目の文は除外し、単語 “`surgery`” を含む文はすべて要約重要度を増やし、単語 “`hypnosis`” を含む文はすべて除外するという意味になります。

したがって、特定の単語または文により高い (またはより低い) 重要性を与えることができます。`summaryConfig` の複数の仕様によって影響を受ける文の重要度は、カスタム・サマリ・アルゴリズムにより解決します。また、このアルゴリズムは、同じ文に適用する仕様の間で重複がある場合に適用します。

- ・ 文 (s) の仕様と単語 (w) の仕様の間で重複がある場合、文の仕様が優先します。
- ・ 2 つの s の仕様または 2 つの w の仕様に関して、包含 (true) と除外 (false) の間で重複がある場合、包含の仕様が優先します。
- ・ 特定の要約の長さ、含める必要のある文の数または除外する必要のある文の数の間で重複がある場合、要約の長さは無視されます。

カスタム・サマリのオプションは、%iKnow.Queries.SourceAPI.GetSummary() および %iKnow.Queries.SourceAPI.GetSummaryForText() メソッドの summaryConfig パラメータにより設定できます。

8.13 ソースのサブセットのクエリ

NLP の[フィルタ](#)を使用すると、クエリにソースを含めたり、クエリからソースを除外できます。以下に基づいて、ソースを包含または除外することができます。

- ・ ソースの[ランダム・サンプリング](#)
- ・ [ソース・コンテンツ](#)：指定されるエンティティまたはディクショナリに一致するエンティティ
- ・ ソースの特性 (メタデータ)：[ソース ID](#)、[文の数](#)、およびソースの[インデックス作成日](#) (ソースが NLP にロードされた日付)
- ・ ソースのユーザ定義の[メタデータ特性](#)

NLP では、論理 AND および論理 OR 演算子により、複数のフィルタを組み合わせで使用できます。詳細は、このドキュメントの "[フィルタ処理](#)" の章を参照してください。

9

意味的属性

InterSystems NLP は、自然言語テキスト内の**コンセプトとそのコンテキストを識別**します。

コンセプトとは、それ自体が意味を持っている不可分の単語グループで、多くの場合、その単語グループが使用されている文に依存しない存在です。例えば、“Patient is being treated for acute pulmonary hypertension” という文では、InterSystems NLP は単語グループである “patient” と “acute pulmonary hypertension” をコンセプトとして識別します。文中では、リレーションが複数のコンセプトを 1 つのパスにリンクすることで、コンセプトどうしの有意な関連を示します。前述の例では、リレーション “is being treated for” によってコンセプトが関連付けられています。

しかし、文 “Patient is not being treated for acute pulmonary hypertension” では、コンセプト “acute pulmonary hypertension” に同じ本質的な意味がありますが、コンテキストは明らかに異なっています。この場合、このリレーションによって否定的な関係が表現された文の一部として、このコンセプトが使用されています。自然言語処理を使用して肺の問題に注目するアプリケーションでは、このコンセプトが使用されている状況と、前述の例での状況は、明らかに異なる方法で扱われます。

InterSystems NLP は、パスとそれを構成する各エンティティのコンテキスト上の意味に影響する意味的属性（否定など）がパスにある場合に、検索用の索引的処理を適用することで、ソース・テキストに関する充実したデータ・セットを提供し、高度な分析を実行できるようにします。

9.1 属性が機能するしくみ：マーカ用語と属性拡張

文中では、意味的属性を通常はマーカ用語で示しています。“Patient is not being treated for acute pulmonary hypertension,” という文では、単語 “not” が否定を表すマーカ用語です。マーカ用語は通常 1 つの単語または複数の単語の組み合わせですが、エンティティ全体である必要はありません。前述の例では、“not” は “is not being treated for” というリレーション・エンティティの一部にすぎません。

ユーザ・ディクショナリを使用して、属性に追加のマーカ用語を指定できます。**構成**の一部としてユーザ・ディクショナリを指定すると、InterSystems NLP ではそのユーザ・ディクショナリで定義したマーカ用語が認識され、適切な**属性拡張**を実行して、文またはパスのどの部分に属性が適用されるかが判断されます。**プログラム**でユーザ・ディクショナリに属性のマーカ用語を追加する場合、`%iKnow.UserDictionary` クラスには各属性タイプに固有のインスタンス・メソッドがあります（`AddPositiveSentimentTerm()` など）。また、このクラスは一般的な属性を定義するための `AddAttribute()` メソッドも提供します。

9.1.1 エンティティ・レベル：マーカ用語のビット・マスク

InterSystems NLP では、エンティティが分析の最小単位なので、エンティティにマーカ用語が単語レベルで使用されていると、ビット・マスクを使用してエンティティ・レベルでアノテーションが付けられます。ビット・マスクは 0 と 1 で構成される文字列です。この文字列での各ビット・マスクの位置は、エンティティを構成する単語のシーケンスにある各単語を表

します。指定された位置のビット・マスクは、対応する単語がマーカ用語であることを示しています。例えば、エンティティ "is not being treated for" には、否定のビットマスク "01000" が割り当てられます。

9.1.2 パスレベル : 拡張属性の位置と範囲

InterSystems NLP は、意味的属性のマーカ用語を含むエンティティにインデックスを作成するにとどまりません。そのほか、InterSystems NLP は文法の理解を活用して属性拡張を実行し、パスの中でマーカ用語前後のすべてのエンティティにフラグを付けます。これらのエンティティは、属性による影響も受けます。文 "Patient is not being treated for acute pulmonary hypertension or CAD, but reports frequent chest pain," では、コンセプトである "acute pulmonary hypertension" と "CAD" に、拡張否定属性の一部としてフラグが付きませんが、コンセプト "frequent chest pain" にはフラグが付きません。

拡張された意味的属性の情報には、次の 2 つのトリガを使用してパス・レベルでアノテーションが付ききます。

- ・ position : この属性により影響を受ける最初のエンティティの場所
- ・ span : この属性により影響を受ける連続するエンティティの数

前述の文では、否定の開始位置は 1 ("Patient") で、範囲は 5 ("CAD" で終了) になります。

属性拡張により、InterSystems NLP は高度な分析を実行するための比類のない能力を提供します。パスレベルで情報を利用できるため、例えばコンセプト "CAD" の肯定的な使用と否定的な使用を容易に識別して、画面上で明確に強調表示することや、アプリケーションに高度な解釈のロジックを実現することなどができます。

9.2 属性データへのアクセス

属性の分析情報は、以下のメソッドで使用できます。

- ・ %iKnow.Queries.SourceAPI: GetAttributes()
- ・ %iKnow.Queries.EntityAPI: GetOccurrenceAttributes() および IsAttributed()
- ・ %iKnow.Queries.PathAPI: GetAttributes()
- ・ %iKnow.Queries.SentenceAPI: GetAttributes() および GetHighlighted()

9.2.1 属性のデータ構造

InterSystems NLP はマーカ用語を識別し、隣接するどのエンティティがその影響を受けるかを特定したうえで、その属性についてのデータを保存します。これにより、前に挙げた %iKnow.Queries パッケージの API のいずれかを使用して、そのデータにアクセスできるようにします。

属性データは %List として保存されます。この属性 %List の正確なコンテンツは、その情報提供先の分析のレベルによって異なります。該当するコンテンツは以下の順序で表示されます。

1. 属性タイプの数値 ID。使いやすさを考慮して、%IKPublic の #include ファイルは、クエリでこれらの値を指定するための名前付きマクロを提供します。例えば、確実性属性タイプの ID は、\$\$\$IKATTCERTAINTY マクロを使用して呼び出すことができます。
2. 属性タイプの名前が使用された文字列 ("negation" や "!measurement" など)。
3. 分析のレベルを示す数値 ID (パスのレベルなど)。使いやすさを考慮して、%IKPublic の #include ファイルは、クエリでこれらの値を指定するための名前付きマクロを提供します。例えば、\$\$\$IKATTLVLPATH マクロを呼び出してパスのレベルを参照できます。

- 属性を持ち、指定された分析レベルにある要素の数値 ID。この中の単語は、0 または 1 を使用した文字列です。
- 要素の中で属性による影響を受ける最初のエンティティの位置。無関係な単語 ("the" や "a") は独立したエンティティとしてカウントされます。例えば、文 "The White Rabbit usually hasn't any time," では、エンティティ 3 の関係 "usually hasn't" に否定マークが付きます。文 "The White Rabbit usually has no time," では、エンティティ 4 の概念 "no time" に否定マークが付きます。
- 属性の範囲。これは、その属性が影響を及ぼす連続するエンティティの数です。例えば、"The man is neither fat nor thin" には、5 つのエンティティ ("man"、"is neither"、"fat"、"nor"、および "thin") で構成する否定範囲があります。
- 該当する場合、この属性に関連付けられたプロパティの名前を使用した文字列。測定のため、この文字列は、測定値とその単位を検出順に記述したコンマ区切りのリストになります。確実性属性の場合、この文字列には確実性レベル c. の数値が記述されます。

9.2.2 例

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、["サンプル・プログラムに関するメモ"](#)を参照してください。

以下の例では、%iKnow.Queries.SourceAPI.GetAttributes() を使用して、ドメイン内の各ソースから否定属性を持つパスと文を検索します。パス ID または文 ID、および各否定の開始位置と範囲が表示されます。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
  GOTO ListerAndLoader }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
  QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeCause FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeCause")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
GetSourcesAndAttributes
SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.srcs,domId,1,numSrcD)
SET i=1
WHILE $DATA(srcs(i)) {
  SET srcId = $LISTGET(srcs(i),1)
  SET i=i+1
  DO ##class(%iKnow.Queries.SourceAPI).GetAttributes(.att,domId,srcId,1,10,"", $$ $IKATTTLVLANY)
  SET j=1
  WHILE $DATA(att(j)) {
    IF $LISTGET(att(j),1)=1 {
      SET type=$LISTGET(att(j),2)
      SET level=$LISTGET(att(j),3)
```

```

SET targId=$LISTGET(att(j),4)
SET start=$LISTGET(att(j),5)
SET span=$LISTGET(att(j),6)
  IF level=1 {WRITE "source ",srcId," ",type," path ",targId," start at ",start," span
",span,!}
  ELSEIF level=2 {WRITE "source ",srcId," ",type," sentence ",targId," start at ",start,"
span ",span,!}
  ELSE {WRITE "unexpected attribute level",! }
}
SET j=j+1
}
}

```

9.3 サポートされる属性

InterSystems NLP ではいくつかの意味的属性タイプがサポートされ、属性タイプごとに独立したアノテーションが付きます。言い換えると、あるエンティティが使用されていると、指定した言語モデルでサポートされている属性タイプの任意の個数と任意の組み合わせに対してアノテーションが付く可能性があります。

英語の場合、InterSystems NLP には、このようなすべての属性タイプ（汎用属性を除く）に対してマーカ用語が用意されています。意味的属性のサポートは属性タイプに応じて異なります。このバージョンの InterSystems NLP で言語モデルごとにサポートされている意味的属性タイプを以下の表に示します。参照しやすいように、各属性タイプの横には、[ドメイン・エクスプローラ](#)および[インデックス作成結果](#)ツールで強調表示に既定で使用される色を括弧で囲んで記述しています。

属性 (強調表示の色)	英語	チェコ語	オランダ語	フランス語	ドイツ語	日本語	ポルトガル語	ロシア語	スペイン語	スウェーデン語	ウクライナ語
否定 (赤)	可	可	可	可	可	可	可	可	可	可	可
時間 (オレンジ)	可	可	可	可	不可	不可	不可	可	不可	可	可
期間 (緑)	可	不可	不可	不可	不可	可	不可	不可	不可	不可	不可
頻度 (黄)	可	不可	不可	不可	不可	可	不可	不可	不可	不可	不可
測定 (ピンク)	可	不可	部分的サポート	不可	不可	可	不可	不可	不可	部分的サポート	不可
感情 (紫)	可	可	可	可	可	不可	可	可	可	可	可
確実性 (黄)	可	不可	不可	不可	不可	不可	不可	不可	不可	不可	不可
汎用 (濃い青)	可	不可	不可	不可	不可	不可	不可	不可	不可	不可	不可

このページの残りの部分では、各属性タイプについて詳しく説明します。

9.4 否定

否定は、肯定文（または文の一部）をその逆の否定に変換するプロセスです。例えば、“I am a doctor.” という文は、“I am not a doctor.” のように否定できます。テキストを分析するときは、多くの場合、あるトピックに関する肯定的なステータス

トメントをそのトピックに関する否定的なステートメントから分離することが重要です。InterSystems NLP は、ソース・インデックスの作成時に、属性 “negation” を文と関連付け、テキストのどの部分が否定されているかを示します。

最も簡単な形式の否定は、“no” または “not” を肯定的なステートメントまたは語句に関連付けるだけです。実際の言語での否定は、より複雑な言語固有の操作です。否定には、以下の 2 つの基本タイプがあります。

- 形式的 (文法的) 否定は、常にテキスト内の特定の形態的要素によって示されます。例えば、“no”、“not”、“don’t”、およびその他の特定の否定用語です。これらの否定要素は、概念 “He has no knowledge of” の一部になったり、関係 “He doesn’t know anything about” の一部になったりします。形式的否定は常に二元的です。文 (または文の一部) は、否定要素を含む (したがって、否定である) か、または肯定的であるかのどちらかになります。
- 意味的否定は、コンテキストに依存する複雑な形式の否定であり、テキスト内の特定の形態的要素によって示されません。意味的否定は、特定のコンテキストにおける単語または単語群の特定の意味に依存するか、または意味と時制の特定の組み合わせ (例えば、ロマンス言語の接続法や仮定法の時制) によって発生します。例えば、“Fred would have been ready if he had stayed awake” や “Fred would have been ready if the need had arisen” は、Fred の心構えについて反対のことを言っています。意味的否定は、二元的な原則ではありません。絶対的であることはほとんどなく、コンテキスト上の洞察や文化的な洞察に左右されます。

InterSystems NLP の言語モデルには、さまざまな言語固有の否定語および否定構造が含まれています。InterSystems NLP では、これらの言語モデルを使用すると、ソースをロードする過程で形式的否定のほとんどのインスタンスを自動的に識別し、分析のためにフラグを付けることができます。ただし、InterSystems NLP は意味的否定のインスタンスを識別できません。

InterSystems NLP では、否定の最も大きい単位はパスです。したがって、パスより大きいテキスト単位で構成された否定を識別できません。文は、全部ではありませんが、その多くが 1 つのパスで構成されます。

9.4.1 否定の特殊なケース

以下は、英語の否定に関する特殊事情です。

- No. : 英語の単語 “No.” (大文字で始まり、ピリオドで終わる。引用符は付く場合と付かない場合がある) は、省略形として扱われます。否定としては扱われず、文の終わりとしても扱われません。小文字の “no.” は否定および文の終わりとして扱われます。
- Nor : 文の先頭の単語 “Nor” は、否定としてマークされません。文の途中にある単語 “nor” は、否定としてマークされます。
- no-one : ハイフンで結ばれた単語 “no-one” は、否定マーカとして扱われます。他のハイフンで結ばれた形式 (“no-where” など) は、否定マーカとして扱われません。
- 偽りの否定 : 形式的否定はコンテキストでなく単語に依存するため、場合によっては偽りの否定が発生することは避けられません。例えば、“There was no answer” と “The answer was no” のどちらの文にも否定のフラグが付けられます。

否定は文の単位で意味を持つので、InterSystems NLP で何が 1 つの文と見なされるか (または見なされないか) を理解することが重要です。InterSystems NLP による文の識別方法の詳細は、“コンセプトの概要” の章の “[InterSystems NLP が識別する論理テキスト・ユニット](#)” を参照してください。

9.4.2 否定とスマート・マッチング

InterSystems NLP は、スマート・マッチング・ディクショナリとのマッチング時に、否定されたエンティティを認識します。否定の一部であるエンティティの数を計算し、その数をマッチ・レベル情報の一部として格納します (これは、`GetMatchesBySource()` などのメソッドから返されるか、`%iKnow.Objects.DictionaryMatch` の `NegatedEntityCount` プロパティとして取得できます)。これにより、否定されたエンティティを一致したエンティティの総数と比較するなど、否定の内容を考慮に入れて一致の結果を解釈するコードを作成できます。

詳細は、“[スマート・マッチング : ディクショナリの使用](#)” を参照してください。

9.5 時間、期間、および頻度

ドキュメントには、時間、期間、または頻度を表す構造化データが含まれる場合もあります。これらのデータには、通常、コンセプトの一部としての属性用語から成る個別の属性としてアノテーションが付けられます。これらの属性は、対象の言語で識別されるマーカ用語に基づいて識別されます。マーカ用語には、特定の数値が含まれる場合と含まれない場合があります。

数字または単語で指定される数値は、ほとんど常に測定属性として扱われます。ただし、時間属性には数値、頻度属性には順序数を含めることができます。

以下は、数値を制御するための英語モデルでのガイドラインの一部を示しています。

- ・ 数字：用語に関連付けられていない数字は測定または年の可能性があります。1900 から 2039 までの数字は年と見なされ、時間属性が割り当てられます ("1923, 2008 applicants")。この範囲以外の数字 ("1776") は、単語 "year" が指定されていない限り ("the year 1776")、年とは見なされません。この範囲以外の単独の数字 ("1776") には属性が割り当てられません。この範囲以外の数字で用語に関連付けられている場合は、測定と見なされます ("1776 applicants")。アポストロフィが付いた 2 桁の数字 ("Winter of '89" など) は年と見なされ、時間属性が割り当てられます。数字または貨幣の句読点が付いた数値 ("1,973"、"-1973"、"1973.0"、または "\$1973") および単語で表された数値 ("nineteen seventy-three") は測定と見なされます。有効な時間の数値 ("12:34:33") には、時間属性が割り当てられます。
- ・ 順序数：コンセプト内で他の単語を伴う序数には、綴られている場合 ("the fourth attempt") は頻度属性が割り当てられ、数字で指定されている場合 ("the 4th attempt") は測定属性が割り当てられます。"tenth" より大きい順序数が綴られている場合は頻度属性は割り当てられません。コンセプト内の順序数に単独で属性が割り当てられることはありません ("a fifth of scotch"、"came in third")。

綴られている順序数の前に別の数値がある場合 ("first through tenth") は、分数 ("one third"、"two fifths") と同様に測定属性が割り当てられますが、"one second" は例外で、期間属性が割り当てられます。

任意の大きさの順序数で月の名前を伴う場合 ("sixteenth of October"、"October 16th"、"October sixteenth") は、時間属性が割り当てられます。ただし、"May" は例外で、英語ではあいまいなため、属性が割り当てられません。

注釈 InterSystems NLP では、チェコ語、オランダ語、ロシア語、スウェーデン語、およびウクライナ語の時間属性をサポートしますが、現在のところ、時間、期間、および頻度に対する個別のアノテーションはサポートしていません。

9.6 測定

ドキュメントには一般的に、数量を表す構造化データ要素が含まれています。これらの要素としては、数、長さ、重さ、貨幣額、投薬量、およびその他の数量的表現を含むことができます。これらの要素は以下のようにさまざまなパターンに従うことができます。

1. 単位を伴う数字 (例："20 マイル" や "5 mg")
2. 数字と単位の両方で構成する文字列 (例："\$200")
3. カウント対象に関連付けられたコンセプトの用語を伴う数字 (例："50 人")
4. 測定値であることを示す表記を伴う数字 (例："血圧：120/80")

InterSystems NLP では、数値と単位の組み合わせ (上記のパターン 1 および 2) に、単語レベルの測定マーカ用語としてアノテーションが付きまます。その他のケース (上記のパターン 3 と 4) では、単語レベルでの測定値としての数字にのみアノテーションが付きまます。

小数の数値を処理するために、測定の数値の一部として先頭にピリオドが追加されます。

数値と単位にアノテーションを付けることに加えて、インターシステムズの NLP は、属性拡張規則を使用して、測定に“含まれる”その他のコンセプトを識別します。アノテーションが付けられたこの一連のコンセプトによって、測定自体だけでなく、測定対象となっているものも把握されます。(上記のパターン 3 と 4 の場合、測定に“含まれる”、測定値であることを示す表記またはコンセプトの用語は範囲の一部になります。)

これらの拡張属性は、以下のことに使用できます。

- ・ ドキュメント内の測定可能なファクトすべてを、強調表示または一覧表示することで抽出します。
- ・ 特定のコンセプトの表示を、特定の測定に関連付けられているものだけに絞り込みます。

注釈 英語および日本語では、数値に関連付けられているすべてのコンセプトに測定属性のフラグが設定されます。例外については、“[時間](#)”、“[期間](#)”、“[および頻度](#)”を参照してください。オランダ語およびスウェーデン語では、関連付けられたコンセプトの用語を伴う数字 (上記のパターン 3) は、測定値としてマークされません。

9.7 感情

感情属性では、肯定的な感情または否定的な感情のどちらかが含まれているものとして、文にフラグを設定します。感情用語は、分析するテキストの種類に大きく依存します。例えば、顧客対象の意識調査におけるコンテキストでは、以下の用語に対して感情属性によるフラグが設定される可能性があります。

- ・ 単語 “avoid”、“terrible”、“difficult”、“hated” は、否定的な感情を伝えます。
- ・ 単語 “attractive”、“simple”、“self-evident”、“useful”、“improved” は、肯定的な感情を伝えます。

感情用語はソース・テキストの内容に固有であることが多いため、InterSystems NLP ではごく少数の感情用語のみが自動的に特定されます。肯定的な感情または否定的な感情の属性を持つものとして、追加の単語にユーザがフラグを設定することはできます。特定の単語に感情属性を指定するには[ユーザ・ディクショナリ](#)を使用します。ソース・テキストをドメインにロードするときに、該当する用語のすべての出現箇所と、その用語の影響を受ける文の部分に、指定した肯定的な感情マーカまたは否定的な感情マーカによるフラグが設定されます。

例えば、否定的な感情属性を持つものとして “hated” が指定されていて、肯定的な感情属性を持つものとして “amazing” が指定されている場合に、それらの属性を InterSystems NLP が以下の文に適用するとします。

I hated the rain outside, but the running shoes were amazing.

否定的な感情は “rain” に影響して、肯定的な感情は “running shoes” に影響します。

肯定的な感情属性または否定的な感情属性が文の否定部分に出現する場合、感情の意味が逆になります。例えば、単語 “good” に肯定的な感情のフラグが設定されている場合、文 “The coffee was good” は肯定的な感情ですが、文 “The coffee was not good” は否定的な感情です。

9.8 確実性

事実の記述は、多くの場合、その正確さに関する話し手の確実性 (または確実性の欠如) を示す用語で修飾されます。

- ・ “clearly (明らかに)”、“definitely (確実に)”、“confident that (確信して)”、“without a doubt (疑いなく)”といった用語は、高いレベルの確実性を表す場合があります。
- ・ “could (もしかして)”、“uncertain (不確実な)”、“quite possibly (おそらく)”、“seem to be (思われる)”といった用語は、低いレベルの確実性を表す可能性があります。

これらの用語とそれらが修飾するパスにインデックスを作成することで、価値の高い分析情報を提供できます。例えば、医療記録でのエンティティ “pulmonary embolus (肺動脈塞栓)” の使用状況を分析して、この状態に対する統計的に効果的な対応戦略を判断する場合、このエンティティが “concern for pulmonary embolus (肺動脈塞栓の懸念あり)” という語句の一部として使用されている患者観察記録は除外しようとするかもしれません。

テキストをドメインにロードする際、InterSystems NLP では、確実性用語の使用箇所それぞれと、その用語による影響を受ける文の部分に、確実性属性マーカによってフラグが設定されます。メタデータとしての各確実性属性フラグは 0 ～ 9 の範囲の整数値 c を受け取ります。この値が高いほど確実性のレベルが高くなります。

ただし、確実性または不確実性を示す用語は、分析するテキストの種類に大きく依存します。例えば、病歴聴取の記録に “the patient has no chance of recovery (患者には回復の見込みがない)” とある場合、“no chance of (見込みがない)” という語句は、明らかに高いレベルの確実性を示しています。しかし、この同じ語句が、スポーツチーム監督の “We have no chance of being defeated (負けるとは思わない)” という発言を引用したニュース記事に使用されている場合は、高い確実性を示していないことも考えられます。

このため、InterSystems NLP は、少数の確実性用語のみを自動的に特定します。このような自動的に検出される確実性フラグには、最小 ($c=0$) または最大 ($c=9$) のいずれかの確実性レベルが割り当てられます。

[ユーザ・ディクショナリ](#)に追加の用語を追加することで、その用語に対する確実性属性を指定できます。

9.9 汎用属性

前述の意味的属性のほか、InterSystems NLP には、カスタム属性を定義できるようにする 3 つの汎用フラグが用意されています。[ユーザ・ディクショナリ](#)で 3 つの汎用属性値 (UDGeneric1、UDGeneric2、または UDGeneric3) のいずれかを用語に割り当てることで、その用語をいずれかの汎用属性のマーカとして指定できます。否定または確実性の場合と同様に、InterSystems NLP では、これらの用語の使用箇所それぞれと、それらの用語による影響を受ける文の部分に、指定した汎用属性マーカによるフラグが設定されます。

10

語幹解析

NLP は、語幹解析をサポートしています。これは、ソースの NLP インデックス作成を実行する際のオプション機能です。語幹解析により、NLP は、いくつかの文法的な形式がある単語の語幹形式を認識できます。例えば、一般に、名詞の語幹は主格の単数形の形式です。動詞の語幹は、一般に、不定詞の形式になります。語幹解析は、エンティティ・レベルで行われます。つまり、NLP は、エンティティに含まれる個別の単語の語幹形式ではなく、エンティティの語幹形式を確立するということです。これは、元のソース・テキストからの、追加レベルの正規化になります。

注釈 語幹解析のサポートにより、NLP インデックス作成の領域要件が大幅に増加します。また、NLP インデックス作成のパフォーマンスにも影響を及ぼします。ドメインに対する語幹解析は、この機能が本当に必要な場合にのみ有効にすることをお勧めします。語幹解析は、ロシア語とウクライナ語のテキスト・ソースに使用することが推奨されています。

語幹解析は、[パス関係のエンティティ](#)に対して実行されません。

現時点では、日本語に対する語幹解析のサポートはありません。

語幹解析により、解析済みエンティティは文法的に無効になることがあります。例えば、エンティティ“two bananas”の語幹は“two banana”になります。実際には、エンティティの語幹形式はインデックス作成されたソースには存在しないため、NLP では、語幹形式と“表現形式”をペアにしています。この表現形式とは、ソース内でインデックス作成されたエンティティとして実際に存在する語幹形式に最も近い（レーベンシュタイン距離で判断）エンティティのことです。インデックス作成されたエンティティとしてソース内に語幹形式が存在する場合は、当然のことながら、表現形式と語幹形式が同じになります。

NLP は、語幹の生成にプラグイン・アーキテクチャを採用することで、サードパーティ製の語幹解析ツールを使用できるようにしています（利用可能な場合）。NLP では、語幹の生成に Hunspell を使用します。そのため、語幹解析が必要になったときには、NLP は `INSTALLDIR/dev/hunspell` ディレクトリで指定言語用の Hunspell affix (.aff) ファイルを検索します。指定された言語用の Hunspell デictionary が用意されていない場合、NLP は、指定された言語用の %Text サブクラスを検索します。InterSystems IRIS® Data Platform は、5 つの NLP サポート対象言語用の %Text データ型クラス、%Text.English（英語）、%Text.French（フランス語）、%Text.German（ドイツ語）、%Text.Portuguese（ポルトガル語）、および %Text.Spanish（スペイン語）を提供しています。Hunspell デictionary ファイルも言語固有の %Text クラスも使用できない場合、NLP は %Text.Text クラスを使用します。詳細は、“インターシステムズ・クラス・リファレンス”の %Text パッケージ・クラスのドキュメントを参照してください。

10.1 語幹解析の構成

語幹解析は、空のドメインに対してのみ有効化または無効化できます。ドメインにソース・テキストが入力されると、最初にドメインのすべてのコンテンツを削除した場合を除き、そのドメインに対する語幹解析オプションは変更できなくなります。

空の NLP ドメインに対して語幹解析を有効化するには、`$$$IKPSTEMMING ドメイン・パラメータ` を 1 に設定します。語幹解析は、既定で無効化されています。空の NLP ドメインの語幹解析を無効化するには、`$$$IKPSTEMMING` を 0 に設定します。

このパラメータを設定するときには、ドメインが空になっている必要があります。語幹解析が有効化されているドメインにソース・テキストを追加すると、NLP は、すべてのエンティティに語幹形式を作成し、語幹をクエリできるよう適切なデータ構造に移入します。

語幹解析機能のインスタンスを作成して、既定の言語を指定します (または `%iKnow.Stemming.MultiLanguageConfig` を使用して複数の言語を指定します)。言語の指定には、その言語の ISO 639-1 の 2 文字の省略形式を使用します。

10.1.1 Hunspell

InterSystems IRIS は、`dev/hunspell` ディレクトリに、ロシア語用 (ru) とウクライナ語用 (uk) の Hunspell ファイルを用意しています。その他の言語用の Hunspell デクシオナリ・ファイルは、InterSystems IRIS の `dev/hunspell` ディレクトリに配置する必要があります。

NLP は、ある重要な一点で、Hunspell 語幹解析の動作を変更しています。Hunspell 語幹解析では、単語から接頭語が削除されます。NLP では、Hunspell によって削除された接頭語を元に戻してから、結果としての語幹形式にインデックスを作成します。

Hunspell が 1 つの単語に対して考えられる複数の語幹を返した場合、NLP は、エンティティが概念であるか、関係であるかを判断するために、エンティティのコンテキストを使用して、そのような選択肢のあいまいさを解消しようとします。

10.2 語幹取得のメソッド

以下に示す `%iKnow.Queries.EntityAPI` のメソッドは、語幹の値を返します。

- `GetStem()` は、指定したエンティティ文字列に対応する語幹文字列を返します。
- `GetStemId()` は、指定した語幹文字列の語幹 ID を返します (ドメイン内に語幹文字列が存在する場合)。
- `GetStemValue()` は、指定した語幹 ID の語幹文字列を返します。
- `GetStemRepresentationForm()` は、指定した語幹 ID の表現形式を返します。

指定した語幹 ID についての頻度と分布を返すこともできます。

10.3 語幹の使用

多くの NLP クエリ・メソッドでは、エンティティではなく、語幹に対してクエリを実行できます。こうしたクエリ・メソッドは、メソッド引数 `pUseStems=1` を設定することで、エンティティの値そのものではなく、語幹形式の値に基づいた値を返します。既定の `pUseStems=0` では、こうしたクエリ・メソッドはエンティティの値そのものに基づいた値を返すようになります。

例えば、既定の `GetTop()` メソッドは、ドメイン内で最も頻繁に出現するエンティティを返します。`pUseStems=1` を指定すると、このメソッドは、ドメイン内で最も頻繁に出現する語幹形式を返します。これは、文法的な形式のみ異なる複数のエンティティを 1 つの語幹形式に統合する可能性もあります。

また、**語義的な近似**の `GetProfile()` メソッドでは、`pUseStems=1` を設定することで、ドメイン内のエンティティではなく、語幹形式の近似プロファイルを返すことができます。

語幹解析をサポートしていないドメインでは、`pUseStems=1` を設定しても、このメソッドは結果を返しません。

詳細は、このドキュメントの **"NLP クエリ"** の章を参照してください。

11

Skiplist

skiplist とは、クエリから返されることを望まないエンティティのリストです。例えば、ソース・テキストにあいさつ文が含まれている場合は、“many thanks” や “best regards” などの実際的な情報を含まない決まり文句を skiplist に入れたいと思うでしょう。また、skiplist を使用して、クエリ結果の分析対象とするには一般的で広く使用されすぎている上位概念を抑制することもできます。

注釈 skiplist を適用したクエリの結果を表示する際は、skiplist を使用すると一部の情報が抑制され、クエリ結果が通知なしで変更されることに注意する必要があります。skiplist は、データ・コンテンツ、クエリ結果を参照するユーザ、およびクエリ結果が表示されるコンテキストに適したものを使用してください。

11.1 skiplist の作成

特定のドメインに割り当てられる skiplist を定義したり、現在のネームスペースの任意のドメインで使用できるドメイン非依存 (ドメイン間共通) の skiplist を定義したりすることができます。skiplist は、以下の 2 つの方法で定義できます。

- ・ InterSystems IRIS [ドメイン・アーキテクト](#)を使用します。このインタフェースを使用して、ドメイン内 skiplist の定義、skiplist エンティティの追加と削除、skiplist の削除、またはドメインに対して定義された全 skiplist のリストができます。このインタフェースでは、エンティティを文字列として指定することによる skiplist 生成をサポートしています。
- ・ `%iKnow.Utils.MaintenanceAPI` クラスのメソッドを使用して、skiplist の定義、生成、および維持を行います。このクラスにより、ドメイン固有の skiplist とドメイン間共通の skiplist の両方を作成できます。このクラスには、文字列としてエンティティを指定するか、エンティティ ID でエンティティを指定して、skiplist にデータを入力するためのメソッドがあります。この章では、skiplist の `%iKnow.Utils.MaintenanceAPI` クラス・メソッドについて、いくつかの使用方法を示します。

`%iKnow.Utils.CopyUtils` クラスの `CopySkipLists()` メソッドを使用すると、ドメイン内のすべての定義済み skiplist を別のドメインにコピーできます。

ドメイン・エクスプローラおよび基本ポータル [ユーザ・インタフェース](#)では、skiplist の使用をサポートしています。

以下の例では、ドメイン固有の skiplist を作成し、その skiplist に要素を追加します。そして、ドメイン用のすべての skiplist、およびこのドメインのすべての要素をリストします。最後に、skiplist を削除します。

ObjectScript

```
DomainCreateOrOpen
SET domn="mydomainwithsl"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
  SET domId=domo.Id }
ELSE { SET domo=##class(%iKnow.Domain).%New(domn)
```



```

        DO domo.%Save()
        SET domId=domo.Id }
CreateSkiplist
    SET slname="AviationSkiplist"
    SET slId=##class(%iKnow.Utills.MaintenanceAPI).CreateSkiplist(domId,slname,
        "Aviation non-mechanical terms skiplist")
PopulateSkiplist
    SET skip=$LB("aircraft","airplane","flight","accident","event","incident","pilot",
        "student pilot","flight instructor","runway","accident site","ground","visibility","faa")

    SET ptr=0
    FOR x=0:1:100 {
        SET moredata=$LISTNEXT(skip,ptr,val)
        IF moredata=1 {
            SET stat=##class(%iKnow.Utills.MaintenanceAPI).AddStringToSkiplist(domId,slId,val)
        }
        ELSE { WRITE x," entities in skiplist",!!
            GOTO ListSkiplist }
    }
ListSkiplist
    SET stat=##class(%iKnow.Utills.MaintenanceAPI).GetSkiplists(.sl,domId,0)
    SET i=1
    WHILE $DATA(sl(i)) {
        WRITE $LISTTOSTRING(sl(i),"",1),!
        SET i=i+1 }
    WRITE "Printed the ",i-1," skiplists",!
    SET stat=##class(%iKnow.Utills.MaintenanceAPI).GetSkiplistElements(.sle,domId,slId)
    /* IF stat=1 {WRITE "success",!}
    ELSE {WRITE "GetSkiplistElements failed" QUIT } */
    SET j=1
    WHILE $DATA(sle(j)) {
        WRITE $LISTTOSTRING(sle(j),"",1),!
        SET j=j+1 }
    WRITE "Printed the ",j-1," skiplist elements",!
CleanUp
    SET stat=##class(%iKnow.Utills.MaintenanceAPI).DropSkiplist(domId,slId)
    IF stat=1 {WRITE "Skiplist deleted",!}
    ELSE {WRITE "DropSkiplist failed" QUIT }

```

CreateSkiplist() メソッドでは、skiplist の名前と説明を指定できます。skiplist 名は、任意の長さの任意の有効文字列を指定できます。skiplist 名では、大文字と小文字が区別されます。skiplist に割り当てる名前は一意にする必要があります。ドメイン固有の skiplist の場合、ドメイン内で一意にする必要があります。ドメイン間共通の skiplist の場合、ネームスペース内で一意にする必要があります。重複した skiplist 名を指定すると、ERROR #8091 が生成されます。skiplist の記述はオプションです。任意の長さの文字列とすることができます。

11.1.1 skiplist とドメイン

作成する各 skiplist は、ドメイン固有にするか、ドメイン間共通 (ドメイン非依存) にして現在のネームスペースの任意のドメインで使えるようにすることができます。

- ドメイン固有の skiplist は、CreateSkiplist() メソッドでドメイン ID を指定することでドメインに割り当てられます。このメソッドは、連続する正の整数として skiplist ID を返します。この skiplist を使用するクエリ・メソッドは、この skiplist ID でこれを参照します。ドメイン固有の skiplist は[語幹解析](#)をサポートできます。
- ドメイン間共通の skiplist はドメインに割り当てられません。代わりに CreateSkiplist() メソッドで 0 のドメイン ID を指定します。このメソッドは、連続する正の整数として skiplist ID を返します。この skiplist を使用するクエリ・メソッドは、負の skiplist ID でこれを参照します。例えば、skiplist ID 8 で識別される skiplist は skiplist ID 値 -8 で参照されます。

ドメイン固有の skiplist を生成する場合、AddEntityToSkiplist() または AddStringToSkiplist() のいずれかを使用できます。ドメイン間共通の skiplist を生成する場合、AddStringToSkiplist() のみ使用できます。

GetSkiplistElements() は、ドメイン間共通の skiplist の entUnId 値について空の文字列を返します。

以下の例では、ドメイン固有の skiplist (AviationTermsSkiplist) とドメイン間共通の skiplist (JobTitleSkiplist) の 2 つの skiplist を作成および生成します。pIncludeCrossDomain ブーリアンが 1 に設定されているため、GetSkiplists() メソッドは両方の skiplist を返します。GetSkiplists() が負の整数としてドメイン間共通の skiplist の skiplist ID を返すことに注意してください。

ObjectScript

```

DomainCreateOrOpen
SET domn="mydomainwithsl"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
  SET domId=domo.Id }
ELSE { SET domo=##class(%iKnow.Domain).%New(domn)
      DO domo.%Save()
      SET domId=domo.Id }
CreateSkipList1
SET slname="AviationTermsSkipList"
SET slId=##class(%iKnow.Utills.MaintenanceAPI).CreateSkipList(domId,slname,
  "Common aviation terms skiplist")
PopulateSkipList1
SET skip=$LB("aircraft","airplane","flight","accident","event","incident","airport","runway")
SET ptr=0
FOR x=0:1:100 {
  SET moredata=$LISTNEXT(skip,ptr,val)
  IF moredata=1 {
    SET stat=##class(%iKnow.Utills.MaintenanceAPI).AddStringToSkipList(domId,slId,val)
  }
}
WRITE "Skiplist ",slname," populated",!
CreateSkipList2
SET sl2name="JobTitleSkipList"
SET sl2Id=##class(%iKnow.Utills.MaintenanceAPI).CreateSkipList(0,sl2name,
  "Aviation personnel skiplist")
PopulateSkipList2
SET jobskip=$LB("pilot","copilot","student pilot","flight instructor","passenger")
SET ptr=0
FOR x=0:1:100 {
  SET moredata=$LISTNEXT(jobskip,ptr,val)
  IF moredata=1 {
    SET stat=##class(%iKnow.Utills.MaintenanceAPI).AddStringToSkipList(0,sl2Id,val)
  }
}
WRITE "Skiplist ",sl2name," populated",!!
ListSkipLists
SET pIncludeCrossDomain=1
SET stat=##class(%iKnow.Utills.MaintenanceAPI).GetSkipLists(.sl,domId,pIncludeCrossDomain)
SET i=1
WHILE $DATA(sl(i)) {
  IF $LIST(sl(i),1)<0 {
    WRITE "cross-domain:",!,$LISTTOSTRING(sl(i),"",1),! }
  ELSE { WRITE "domain-specific:",!,$LISTTOSTRING(sl(i),"",1),! }
  SET i=i+1 }
WRITE "Printed the ",i-1," skiplists",!!
CleanUp
SET stat=##class(%iKnow.Utills.MaintenanceAPI).DropSkipList(domId,slId)
IF stat=1 {WRITE "domain skiplist deleted",!}
ELSE {WRITE "first DropSkipList failed" }
SET stat=##class(%iKnow.Utills.MaintenanceAPI).DropSkipList(0,sl2Id)
IF stat=1 {WRITE "cross-domain skiplist deleted",!}
ELSE {WRITE "second DropSkipList failed" }

```

11.2 skiplist をサポートするクエリ

以下のクエリ・メソッドでは、skiplist を指定するためのパラメータが用意されています。複数の skiplist をそれらのメソッドのいずれかに対して指定するには、[\\$LISTBUILD](#) 関数の使用により、skiplist ID を %List 構造の要素として指定することができます。正の整数の skiplist ID 値としてドメイン固有の skiplist を指定し、負の整数の skiplist ID 値としてドメイン間共通の skiplist を指定します。

エンティティ・クエリ：

- ・ %iKnow.Queries.EntityAPI.GetByFilter()
- ・ %iKnow.Queries.EntityAPI.GetBySource()
- ・ %iKnow.Queries.EntityAPI.GetCountByDomain()
- ・ %iKnow.Queries.EntityAPI.GetCountBySource()
- ・ %iKnow.Queries.EntityAPI.GetNewBySource()

- ・ %iKnow.Queries.EntityAPI.GetRelated()
- ・ %iKnow.Queries.EntityAPI.GetRelatedById()
- ・ %iKnow.Queries.EntityAPI.GetSimilar()
- ・ %iKnow.Queries.EntityAPI.GetSimilarCounts()
- ・ %iKnow.Queries.EntityAPI.GetTop()

文クエリ：

- ・ %iKnow.Queries.SentenceAPI.GetNewBySource()

ソース・クエリ：

- ・ %iKnow.Queries.SourceAPI.GetSimilar()：類似エンティティの選択時および類似スコアの計算時、NLP は skiplist のエンティティを無視します。

11.2.1 skiplist クエリの例

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の例では、対象にするには一般的すぎる非機械的航空用語を抑制しています。ここでは、CreateSkipList() を使用して skiplist を作成し、AddStringToSkipList() を使用してエンティティを skiplist に追加してから、その skiplist を GetTop() メソッドに指定しています。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
CreateSkipList1
    SET slname="AviationTermsSkiplist"
    SET slId=##class(%iKnow.Utills.MaintenanceAPI).CreateSkipList(domId,slname,
      "Common aviation terms skiplist")
PopulateSkiplist
    SET skip=$LB("aircraft","airplane","flight","accident","event","incident","pilot","airport",
      "student pilot","flight instructor","runway","accident site","ground","visibility","faa")

    SET ptr=0
    FOR x=0:1:100 {
      SET moredata=$LISTNEXT(skip,ptr,val)
      IF moredata=1 {
        SET stat=##class(%iKnow.Utills.MaintenanceAPI).AddStringToSkipList(domId,slId,val)
      }
    }
    WRITE "Skiplist ",slname," populated",!

QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
```

```

SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat != 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat != 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
SourceCountQuery
SET numSrcD=##class(%iKnow.Queries.SourceQAPI).GetCountByDomain(domId)
WRITE "The domain contains ",numSrcD," sources",!
TopEntitiesQuery
DO ##class(%iKnow.Queries.EntityAPI).GetTop(.result,dmId,1,20,"",0,0,0,0,$LB(slId))
WRITE "NOTE: the ",slname," skiplist",!,
      "has been applied to this list of top entities",!
SET i=1
WHILE $DATA(result(i)) {
    SET outstr = $LISTTOSTRING(result(i),"",1)
    SET entity = $PIECE(outstr,"",2)
    SET freq = $PIECE(outstr,"",3)
    SET spread = $PIECE(outstr,"",4)
    WRITE "[",entity,"] appears ",freq," times in ",spread," sources",!
    SET i=i+1 }
WRITE "Printed the top ",i-1," entities"

```


12

ソースのフィルタ処理

フィルタを使用して、NLP クエリに供給されるソースを含めたり除外したりすることができます。ドメイン内にロードされたすべてのソースに対してクエリを実行する必要がある場合も多くあります。フィルタにより、フィルタ条件を満たすソースのみにクエリ範囲を制限することが可能です。フィルタは、ソース内にあるエンティティ、またはソース自体に関連付けられた一部の情報（メタデータ）のいずれかに基づいてソースを選択します。フィルタは常にソース全体の包含または除外を行って、フィルタを使用する各クエリ内にて指定されます。

12.1 サポートされるフィルタ

NLP は事前定義されたフィルタを多く備えているほか、ユーザが独自のフィルタを簡単に定義できる機能を装備しています。

- ・ **ソース ID** : SourceIdFilter および ExternalIdFilter では、ソースの ID に基づいてソースを選択できます。NLP はソース・インデックス作成プロセスの一部としてこれらの値を割り当てます。
- ・ **ランダム・ソース** : RandomFilter では、ドメインのソースのランダム・サンプルを選択できます。このサンプルは、ソースの整数または合計ソースのパーセンテージとして指定することができます。
- ・ **ソース・コンテンツ** : SentenceCountFilter では、ソース内の文の最小数や最大数に基づいてソースを選択できます。NLP はインデックス作成プロセスの一部として、ソース内の文をカウントします。
- ・ **エンティティの一致** : NLP では、各ソースにあるエンティティ（概念と相関）に基づいてソースを選択可能なフィルタをいくつか用意しています。DictionaryMatchFilter では、ソース・コンテンツとディクショナリが一致する最小数や最大数に基づいたソースのフィルタ処理が可能です（このフィルタは非推奨の SimpleMatchFilter に代わるものです）。DictionaryTermMatchFilter および DictionaryItemMatchFilter では、同種のディクショナリ・マッチングを使用してソースのフィルタ処理が可能ですが、ディクショナリ全体ではなく、ディクショナリのコンポーネントに対して一致セットを制限します。これらのディクショナリ・フィルタでは、\$\$\$IKPMATSTANDARDIZEDFORM ドメイン・パラメータが指定された場合に、オプションで標準化形式マッチングを実行できます。

ContainsEntityFilter では、（ディクショナリのエンティティ定義ではなく）エンティティのリストを直接指定して、ソースのフィルタ処理を行うことができます。また、ContainsEntityFilter では、リストされたエンティティと類似したエンティティで、ソースのフィルタ処理を任意で行うこともできます。ContainsRelatedEntitiesFilter では、関連付けする必要のある 2 つ以上のエンティティのリストを指定して、ソースのフィルタ処理を行うことができます。つまり、それらのエンティティは同じパス（既定）または同じ CRC のいずれかで出現する必要があります。また、ContainsRelatedEntitiesFilter では、リストされたエンティティと類似した関連エンティティで、ソースのフィルタ処理を任意で行うこともできます。

- ・ **インデックス作成日メタデータ** : NLP では、すべてのソースに対して 1 つのメタデータ・フィールド（DateIndexed フィールド）が自動で提供されます。NLP はソース・インデックス作成プロセスの一部として、このフィールド値を割り

当てます。SimpleMetadataFilter の使用により、このフィールドでは、NLP によってインデックスが作成された日付と時刻に基づいてソースを選択できます。

- ・ **ユーザ定義メタデータ** : NLP では、SimpleMetadataFilter を使用することで、ソースに関連付けたデータ値に基づいてソースを選択するフィルタを定義できます。
- ・ **SQL クエリ** : SqlFilter では、SQL クエリの結果に基づいてソースを選択できます。

GroupFilter を使用すれば、定義するフィルタの結果を論理的に組み合わせることができます。

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

12.2 ソースの ID によるフィルタ処理

最も基本的なソース・フィルタを使用して、フィルタ処理後の結果セットに含めたい各ソースのソース ID または外部 ID を提供することにより、クエリに供給するソースを制限します。

12.2.1 外部 ID でフィルタ処理

ExternalIdFilter には、**外部 ID** が %List 構造にリストされているソースが含まれます。このリスト内の有効な外部 ID でない要素や重複する外部 ID といった要素は、通知なしで無視されます。

以下の例では、ソースを外部 ID でフィルタ処理します。Aviation.Event ソースの外部 ID は単語 “Accident” または “Incident” のいずれかを含んでいます。このフィルタは、外部 ID が単語 “Incident” を含むソースのみを含んでいます。それから、フィルタ処理されたソースの詳細をリストします。

ObjectScript

```
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          GOTO SetEnvironment }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { GOTO SetEnvironment }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT}
SetEnvironment
    SET domId=domoref.Id
    IF ##class(%iKnow.Configuration).Exists("myconfig") {
        SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
    ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),",",1)
          DO cfg.%Save() }
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DefineExtIdFilter
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,100)
    SET i=1
    SET extlist=$LB("")
    WHILE $DATA(result(i)) {
```



```

        SET extId = $LISTGET(result(i),2)
        IF $PIECE(extId,":",3)="Incident" {
            SET extlist=extlist_$LB(extId) }
        SET i=i+1
    }
    SET filt=##class(%iKnow.Filters.ExternalIdFilter).%New(domId,extlist)
SourceCountQuery
    SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
    WRITE "The ",dname," domain contains ",numSrcD," sources",!
ApplyExtIdFilter
    SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt)
    WRITE "The Id filter includes ",numSrcFD," sources:",!
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,100,filt)
        SET j=1
        WHILE $DATA(result(j)) {
            SET intId = $LISTGET(result(j),1)
            SET extId = $LISTGET(result(j),2)
            WRITE intId," ",extId,!
            SET j=j+1
        }
    WRITE "End of list"

```

12.2.2 ソース ID でフィルタ処理

SourceIdFilter には、**ソース ID** が %List 構造にリストされているソースが含まれます。ソース ID は整数値です。これらは任意の順序でリストできます。このリスト内の有効なソース ID でない要素や重複するソース ID といった要素は、通知なしで無視されます。

以下の例では、ソースとして SQL レコードを使用して、複数のソースをソース ID でフィルタ・インします。このデータ・セットのソース ID は 1 から 100 までの数字となります。SourceIdFilter では、5 つのソース ID の包含を指定していますが、それらのソース ID の内で 3 つだけがテーブルのレコードに対応しています。したがって、フィルタ処理されたソース数は合計で 3 になります。

ObjectScript

```

DefineAFilter
    SET srclist=$LB(10,14,74,110,2799)
    SET filt=##class(%iKnow.Filters.SourceIdFilter).%New(domId,srclist)
SourceCounts
    SET numsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
    WRITE "The ",dname," domain contains ",numsrc," sources",!
    SET numfsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt)
    WRITE "Source count after source Id filtering: ",numfsrc

```

iKnow.Filters.Filter クラスは **%iKnow.Filters.SourceIdRangeFilter** サブクラスを含んでいます。以下の例では、ソースの範囲をソース ID でフィルタ処理します。これは、ソース ID が 5 以上 10 以下であるソースを返します。

ObjectScript

```

SET filt=##class(%iKnow.Filters.SourceIdRangeFilter).%New(domId,5,10)

```

12.3 フィルタ処理によるソースからのランダムな選択

%iKnow.Filters.RandomFilter を使用して、ソースのランダム・サンプルを選択できます。ランダム・サンプルでは、ソースの管理可能サブセットに対するテストが可能になります。また、ソース（またはそれらのサブセット）を“トレーニング”および“テスト”のセットに分割することもできます。その場合は、“トレーニング”セットを使用して、NLP 分析（ディクショナリの一致、ソースのカテゴリなど）を定義してから、“テスト”セットを使用して、それらの分析がどの程度他のデータ・セットに当てはまるか判定することになります。これにより、特定のデータ・セットに対する分析の“重複適合”を防ぐことができます。

以下の 2 つの方法で、ランダム・サブセットのサイズを指定できます。

- ・ パーセンテージとして: パーセンテージを (0 ~ 1 の範囲の小数で) 指定すると、このフィルタは指定ドメイン (またはドメインのフィルタ処理されたサブセット) 内のインデックス付けされたソースに関するパーセンテージを返します。例えば、“.5”と指定すると、ドメイン内のソースの 50% がフィルタ結果に含まれます。半分は切り上げられるため、5 つのソースの 50% では 3 つのソースになります。100% は、適切な数値の小数桁を付けて、“.999”と指定します。このフィルタは必要な数のソースをランダムに選択します。
- ・ 整数として: 整数を指定すると、このフィルタは指定ドメイン (またはドメインのフィルタ処理されたサブセット) 内のインデックス付けされたソースの数を返します。例えば、値を “7” にすると、ドメイン内の 7 つのソースがフィルタ結果に含まれます。このフィルタは指定された数のソースをランダムに選択します。

以下の例では、50 個のソースの内 33% をランダムに選択して、17 個のソースを返します。このサンプルを繰り返し実行すると、異なるソースがランダムにサンプリングされるのを確認できます。

ObjectScript

```
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
  ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
    DO domoref.%Save()
    GOTO SetEnvironment }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { GOTO SetEnvironment }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
    QUIT }
SetEnvironment
  SET domId=domoref.Id
  IF ##class(%iKnow.Configuration).Exists("myconfig") {
    SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
  ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),",",1)
    DO cfg.%Save() }
ListerAndLoader
  SET domId=domoref.Id
  SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
  SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
  SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
  IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
  SET stat=myloader.ProcessBatch()
  IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DefineAFilter
  SET filt=##class(%iKnow.Filters.RandomFilter).%New(domId,.33)
SampledSourceQueries
  SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
  WRITE "The ",dname," domain contains ",numSrcD," sources",!
  SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt)
  WRITE "Of these ",numSrcD," sources ",numSrcFD," were sampled:",!
  DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,20,filt)
  SET i=1
  WHILE $DATA(result(i)) {
    SET intId = $LISTGET(result(i),1)
    SET extId = $LISTGET(result(i),2)
    WRITE "sample #",i," is source ",intId," ",extId,!
    SET i=i+1 }
  WRITE "End of list"
```

以下の例では、ドメインのソースをソース ID によりフィルタ処理して、11 個のソースを返します。さらに、ランダム・フィルタの定義時において、このソースに ID フィルタを指定します。したがって、ランダム・フィルタはこれらのソース ID でフィルタ処理されたソースの内の 3 つを返します。このサンプルを繰り返し実行すると、異なるソースがランダムにサンプリングされるのを確認できます。

ObjectScript

```

DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          GOTO SetEnvironment }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { GOTO SetEnvironment }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT }
SetEnvironment
    SET domId=domoref.Id
    IF ##class(%iKnow.Configuration).Exists("myconfig") {
        SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
    ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),",",1)
          DO cfg.%Save() }
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DefineSourceIdFilter
    SET srclist=$LB(1,3,5,7,9,11,13,15,17,21,23)
    SET idfilt=##class(%iKnow.Filters.SourceIdFilter).%New(domId,srclist)
SourceCounts
    SET numsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
    WRITE "The ",dname," domain contains ",numsrc," sources",!
    SET numfsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,idfilt)
    WRITE "Source count after source Id filtering: ",numfsrc,!
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,20,idfilt)
    SET i=1
    WHILE $DATA(result(i)) {
        SET intId = $LISTGET(result(i),1)
        WRITE intId," "
        SET i=i+1 }
    WRITE !,"End of list",!
DefineRandomFilter
    SET rfilt=##class(%iKnow.Filters.RandomFilter).%New(domId,3,idfilt)
RandomSample
    SET numrsrc=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,rfilt)
    WRITE "From ",numfsrc," sources ",numrsrc," are randomly sampled:",!
    DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,20,rfilt)
    SET j=1
    WHILE $DATA(result(j)) {
        SET intId = $LISTGET(result(j),1)
        WRITE intId," "
        SET j=j+1 }
    WRITE !,"End of list",!

```

12.4 文の数によるフィルタ処理

NLP では、ソース・テキストが文に分割されます。以下の例は、文の数が 75 未満のソースをフィルタにより除外します。ソースの合計数を返してから、フィルタを使用して 75 個以上の文を含むソースの合計数を返し、次に再度フィルタを使用して、これらの各ソースの文の数を返します。

ObjectScript

```

DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)

```

```

        GOTO DeleteOldData }
ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      GOTO SetEnvironment }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO SetEnvironment }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT}
SetEnvironment
SET domId=domoref.Id
IF ##class(%iKnow.Configuration).Exists("myconfig") {
  SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),"",1)
      DO cfg.%Save() }
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DefineAFilter
SET filt=##class(%iKnow.Filters.SentenceCountFilter).%New(domId)
SET nsent=75
DO filt.MinSentenceCountSet(nsent)
SourceSentenceQueries
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The domain contains ",numSrcD," sources",!
SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt)
WRITE "Of these ",numSrcD," sources ",numSrcFD," contain ",nsent," or more sentences:",!
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,50,filt)
SET i=1
WHILE $DATA(result(i)) {
  SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
  SET intId = $LISTGET(result(i),1)
  SET extId = $LISTGET(result(i),2)
  WRITE "source ",intId," ",extId
  WRITE " has ",numSentS," sentences",!
  SET i=i+1 }
WRITE i-1," sources listed"

```

文の最小数と最大数の両方を使用してフィルタするには、両方のインスタンス・メソッドを呼び出します。以下のフィルタは、10 個以上 25 個以下の文を含むソースを選択します。

ObjectScript

```

MinMaxFilter
SET min=10
SET filt=##class(%iKnow.Filters.SentenceCountFilter).%New(domId)
DO filt.MinSentenceCountSet(min)
DO filt.MaxSentenceCountSet(min+15)

```

12.5 エンティティ一致によるフィルタ処理

以下のエンティティ・フィルタが用意されています。

- ContainsEntityFilter：エンティティの指定リストを使用して、ソースをフィルタ処理しますが、少なくとも 1 つのエンティティがソース内で出現する必要があります。また、ContainsEntityFilter では、リストされたエンティティと類似したエンティティで、ソースのフィルタ処理を任意で行うこともできます。
- ContainsRelatedEntitiesFilter：関連付けする必要のある 2 つ以上のエンティティの指定リストを使用して、ソースのフィルタ処理を行います。つまり、それらのエンティティは同じパス（フィルタの既定）または同じ CRC のいずれかで出現する必要があります。また、ContainsRelatedEntitiesFilter では、リストされたエンティティと類似した関連エンティティで、ソースのフィルタ処理を任意で行うこともできます。

- DictionaryMatchFilter : エンティティのリストを含む 1 つ以上のディクショナリを使用して、ソースのフィルタ処理を行います。既定により、少なくともそれらのエンティティの内の 1 つは、ソース内に出現する必要があります。必要に応じて、それらのエンティティ一致について指定した最小数が、選択されるソースに対して出現する必要があります。また、ディクショナリ・マッチングでは、\$\$\$IKPMATSTANDARDIZEDFORM [ドメイン・パラメータ](#) が現在のドメインで指定された場合、標準化形式マッチングもサポートします。

12.5.1 ディクショナリー一致によるフィルタ処理

%iKnow.Filters.DictionaryMatchFilter クラスにより、1 つ以上の[ユーザ定義ディクショナリ](#)のコンテンツに基づいて、ソースを選択することができます。また、NLP では、ディクショナリ用語のリストに対する(%iKnow.Filters.DictionaryTermMatchFilter)、またはディクショナリ項目のリストに対する(%iKnow.Filters.DictionaryItemMatchFilter) マッチングによるフィルタ処理もサポートしています。

以下の簡単な例では、2 番目のパラメータにより、ディクショナリ 1 つのみの適用が指定されていますが、複数のディクショナリも %List の要素として指定できます。3 番目のパラメータが既定の 1 に設定されています。つまり、任意のディクショナリ項目の単独一致により、包含するためのソースを選択しています。これをさらに高く設定して、ディクショナリの項目が膨大にあることにより、または膨大なテキストを含むソースのクエリにより、単一のディクショナリー一致が意味のあるものではなく、偶然の産物となる可能性があるソース選択を避ける必要性が生じる場合もあります。4 番目のパラメータは既定値(-1)を採用することで、最大一致数制限をなくしています。最大パラメータが最少パラメータより小さい場合、ディクショナリー一致に関係なく、すべてのソースがフィルタにより選択されます。5 番目のパラメータも既定値を採用していますが、マッチングは一致スコアではなく、一致の数に基づいています。6 番目のパラメータは ensureMatched フラグです。ここでは ensureMatched=2 なので、フィルタのインスタンス化により、フィルタ呼び出しのたびに使用される静的な一致結果を生成します。これを使用することをお勧めします。フィルタのインスタンス化後にディクショナリが変更された場合、ensureMatched を 1 に設定する必要があります。ensureMatched=1 では、毎回フィルタが呼び出される前にマッチングを行うことで、ディクショナリ・コンテンツの変更が考慮されます。ただし、ensureMatched=1 を使用すると処理速度が大幅に落ちてしまうおそれがあります。

ObjectScript

```
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          GOTO SetEnvironment }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { GOTO SetEnvironment }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT }
SetEnvironment
    SET domId=domoref.Id
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
CreateDictionary
    SET dictname="EngineTerms"
    SET dictdesc="A dictionary of aviation engine terms"
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname,dictdesc)
    IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
                  GOTO ResetForNextTime }
    ELSE {WRITE "created dictionary ",dictId,!}
PopulateDictionaryItem1
    SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(domId,dictId,
```



```

    "engine parts",domId_dictId_1)
SET term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "piston")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "cylinder")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "crankshaft")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "camshaft")
DefineAFilter
SET filt=##class(%iKnow.Filters.DictionaryMatchFilter).%New(domId,$LB(dictId),1,,2)
SourceCountQuery
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The ",dname," domain contains ",numSrcD," sources",!
SourcesFilteredByDictionaryMatch
SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt)
WRITE "Of these ",numSrcD," ",numSrcFD," match the ",dictname," dictionary:",!!
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,50,filt)
SET i=1
WHILE $DATA(result(i)) {
    SET intId = $LISTGET(result(i),1)
    SET extId = $LISTGET(result(i),2)
    WRITE "dictionary matches ",intId," ",extId,!
    SET i=i+1 }
WRITE !,i-1," sources included by dictionary match",!
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "deleted dictionary ",dictId,!}
ELSE { WRITE "DropDictionary error ",$System.Status.DisplayError(stat) }

```

12.6 インデックス作成日メタデータによるフィルタ処理

すべての NLP ソースには、DateIndexed メタデータ・フィールドが割り当てられます。このフィールドの値は、NLP によってソースのインデックスが作成された日付と時刻になり、協定世界時形式 (UTC) で \$HOROLOG 形式で示されます。これは \$ZTIMESTAMP の時刻と同じになりますが、DateIndexed には秒数の小数部は含まれません。

DateIndexed を使用してフィルタを作成し、NLP がソースをロードした日時に基づいてソースを含めたり除外したりすることができます。特定の日付と時刻を使用してフィルタ処理したり、特定の日付 (その日付内のすべての時刻値を含む) についてフィルタ処理できます。また、BETWEEN ロジックを使用すると、日付の範囲についてフィルタ処理できます。

以下の例では、今日ロードされたソースについてフィルタ処理しています。

ObjectScript

```

DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      GOTO SetEnvironment }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO SetEnvironment }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT }
SetEnvironment
SET domId=domoref.Id
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()

```



```

IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DefineAFilter
  SET tday = $PIECE($ZTIMESTAMP, ",", 1)
  SET filt=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId, "DateIndexed", "=", tday)
DateIndexedValue
  WRITE "Today is ", $PIECE($ZTIMESTAMP, ",", 1), !
  DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result, domId, 1, 50)
  SET i=1
  WHILE $DATA(result(i)) {
    SET srcId = $LISTGET(result(i), 1)
    SET extId = $LISTGET(result(i), 2)
    SET idate = ##class(%iKnow.Queries.MetadataAPI).GetValue(domId, "DateIndexed", extId)
    WRITE "Source ", srcId, " was indexed ", idate, !
    SET i=i+1 }
SourceSentenceQueries
  SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
  WRITE "The ", dname, " domain contains ", numSrcD, " sources", !
  SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId, filt)
  WRITE "Of these sources ", numSrcFD, " were indexed today"

```

12.7 ユーザ定義メタデータによるフィルタ処理

NLP におけるデータとは、NLP が処理してインデックスを作成するソースのコンテンツです。また、NLP におけるメタデータとは、NLP がインデックスを作成するデータ以外の、ソースに関連付けられた任意のデータです。NLP メタデータを使用して、NLP データを識別します。メタデータ・フィルタはメタデータ・フィールドの値を使用して、クエリに供給するソースを判別します。

注釈 NLP の“メタデータ”の定義では、データ本来の性質ではなく、データの使用方法を説明します。この概念は、InterSystems IRIS® Data Platform ソフトウェアの他の部分でメタデータという言葉が意味するものとは多少異なります。

NLP は、クエリ API とは独立した、既定のメタデータ管理システムを提供します。`%iKnow.Queries.MetadataAPI` クラスと付随する `%iKnow.Filters.SimpleMetadataFilter` は、基本的なメタデータ・フィルタの実装を提供します。カスタムのメタデータ API を実装したい場合は、(最低でも) `%iKnow.Queries.MetadataAPI` インタフェースを実装し、次のように“MetadataAPI”ドメイン・パラメータとしてクラスを登録します：DO

```
domain.SetParameter("MetadataAPI", "Your.Metadata.Class").
```

後述の例では `%iKnow.Filters.SimpleMetadataFilter` クラスを使用しています。

InterSystems SQL では、SQL テーブルの各レコードが 1 つの NLP ソースを構成します。ProcessList() メソッド (レコードが少数の場合) または AddListToBatch() メソッド (レコードが多数の場合) を使用して、リスタ・パラメータを定義します。

- NLP 外部 ID のコンポーネントとして RowID フィールドを定義します。NLP はまた、各行のソース ID を一意の整数値で生成します。この NLP ソース ID は、RowId や他の SQL 識別子の値からは完全に独立しています。
- NLP データとしてインデックスを作成するデータ・フィールドとして、テキスト文字列を含む 1 つまたは複数のフィールドを定義します。
- NLP メタデータ・フィールドとして 1 つまたは複数のフィールドを定義します。NLP はこのメタデータ・フィールドの値を使用して、NLP クエリのソースを選択できます。

データ・フィールドの 1 つとメタデータ・フィールドの両方に同じフィールドを指定できることに注意してください。また、オプションとして、メタデータ・フィールドに対応する metakey フィールドも定義できます。

これを、以下の例に示します。Aviation.Event テーブルは NarrativeFull テキスト・フィールドの他、各種のフィールドを含んでいます。この例では、InjuriesTotal がメタデータ・フィールドとして使用されています。メタデータ・フィールドは 3 つのフィルタで使用されています。その中の 2 つの等値フィルタは InjuriesTotal>2 と InjuriesTotal=3 についてフィルタし、BETWEEN フィルタは InjuriesTotal が 3 以上 5 以下のものをフィルタします。引数なしの DropData() はメタデータを削除しないので、この例では DropData(1) メソッドを使用しています。また、データのリストとロードを行う前に AddField() メソッドを呼び出す必要があることに注意してください。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
        { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
          GOTO DeleteOldData }
    ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
          DO domoref.%Save()
          GOTO SetEnvironment }
DeleteOldData
    SET stat=domoref.DropData(1)
    IF stat { GOTO SetEnvironment }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
          QUIT}
SetEnvironment
    SET domId=domoref.Id
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT TOP 100 ID AS UniqueVal,Type,NarrativeFull,InjuriesTotal,InjuriesTotalFatal FROM
    Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
    SET metaflds=$LB("InjuriesTotal","InjuriesTotalFatal")
AddMetaFields
    SET val=##class(%iKnow.Queries.MetadataAPI).AddField(domId,"InjuriesTotal",
    $LB("=", "<", ">", "BETWEEN"), $$$MDDTNUMBER)
    SET val=##class(%iKnow.Queries.MetadataAPI).AddField(domId,"InjuriesTotalFatal",
    $LB("=", "<", ">", "BETWEEN"), $$$MDDTNUMBER)
UseListerAndLoader
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds,metaflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
CountSources
    SET numsrc=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
ApplyFilter
    SET filt2=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId,"InjuriesTotal",
    ">", 2)
    SET numSrcF2=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt2)
    WRITE "Of these ", numsrc, " sources ", numSrcF2, " had three or more injuries", !
    SET filt3=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId,"InjuriesTotal",
    "=", 3)
    SET numSrcF3=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filt3)
    WRITE "Of these ", numsrc, " sources ", numSrcF3, " had three injuries", !
    SET filtb=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId,"InjuriesTotal",
    "BETWEEN", "3;5")
    SET numSrcFb=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,filtb)
    WRITE "Of these ", numsrc, " sources ", numSrcFb, " had between 3 and 5 injuries", !

```

12.7.1 メタデータ・フィルタ演算子

それぞれのフィルタに 1 つ以上の等値演算子を割り当てます。フィルタを文字列値とマッチングする場合は、“=” 等値演算子を使用します。フィルタを数値とマッチングする場合は、“=”、“<”、“<=”、“>”、“>=” のうち 1 つ以上の演算子を使用できます。等値演算子は常に、リスト構造の引用符付きの文字列要素として指定します。等値演算子は 1 つの値に対してマッチングされます。詳細は、以下の例を参照してください。

ObjectScript

```
SET filt=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId,metafldname,"=",today)
```

BETWEEN 演算子は、\$\$\$MDValseparator (セミコロン文字) で区切られる値のペアを含むパラメータ文字列に対してマッチングされます。詳細は、以下の例を参照してください。

ObjectScript

```
SET filt=##class(%iKnow.Filters.SimpleMetadataFilter).%New(domId,metafldname,
    "BETWEEN","yesterday;tomorrow")
```

12.8 SQL クエリによるフィルタ

%iKnow.Filters.SqlFilter クラスでは、SQL クエリの結果に基づいて SQL ソースを選択できます。このクエリでは、以下のフィールドのいずれかを選択できます。

- ・ SourceId : 選択されたソースの (内部の) ソース ID
- ・ ExternalId : 選択されたソースの完全な外部 ID
- ・ IdField および GroupField : ドメインにソースを追加する際に識別子として共に使用される 2 つの列。つまり、ローカル参照 (IdField) およびグループ名 (GroupField)。%iKnow.Source.SQL.Lister も参照してください。

これらの結果列名では、大文字と小文字が区別されることに注意してください。

例えば、以下のフィルタでは、6 番目に取得されたソースの SourceId が選択されます (この場合、SourceId 45)。

ObjectScript

```
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  GOTO SetEnvironment }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO SetEnvironment }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
  QUIT}
SetEnvironment
SET domId=domoref.Id
IF ##class(%iKnow.Configuration).Exists("myconfig") {
  SET cfg=##class(%iKnow.Configuration).Open("myconfig") }
ELSE { SET cfg=##class(%iKnow.Configuration).%New("myconfig",0,$LISTBUILD("en"),",",1)
  DO cfg.%Save() }
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseListerAndLoader
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,50)
FilterSources
SET filter=##class(%iKnow.Filters.SqlFilter).%New(domId,
  "SELECT '$_$LIST(result(6),1)_' AS SourceId")
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.fresult,dmId,0,0,filter)
WRITE !,"Filtered results:",!
SET j=1
WHILE $DATA(fresult(j)) {
  WRITE $LISTTOSTRING(fresult(j)),!
  SET j=j+1 }

```

以下のフィルタでは、ExternalId でソースが選択されます。

ObjectScript

```
SET filter=##class(%iKnow.Filters.SqlFilter).%New(domId,
  "SELECT '$_$LIST(result(1),2)_' AS ExternalId")
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(domId,0,0,filter)
ZWRITE result

```

12.9 フィルタ・モード

FilterMode 引数は、フィルタ適用後に実行する必要がある統計上の再処理を指定します。再処理を実行しない場合、フィルタは適用されますが、頻度および分散統計はフィルタ処理前に項目について計算された値になり、並べ替え順序は前と同じになります。使用できるフィルタ・モードを以下に示します。

FilterMode	整数コード	フィルタ処理	頻度の再計算	分散の再計算	結果を再度並べ替え
\$\$\$FILTERONLY	1	あり	なし	なし	なし
\$\$\$FILTERFREQ	3	あり	あり	なし	なし
\$\$\$FILTERSPREAD	5	あり	なし	あり	なし
\$\$\$FILTERALL	7	あり	あり	あり	なし
\$\$\$FILTERFREQANDSORT	11	あり	あり	なし	あり
\$\$\$FILTERSPREADANDSORT	13	あり	なし	あり	あり
\$\$\$FILTERALLANDSORT	15	あり	あり	あり	あり

既定値は \$\$\$FILTERONLY です。

\$\$\$ マクロの使用方法は、“NLP の実装” の章の “定数” を参照してください。

12.10 GroupFilter の使用による複数フィルタの組み合わせ

NLP では **GroupFilter** クラスを提供することで、ロジックを指定して、他のフィルタの結果を組み合わせることを可能にしています。最初は、定義されたロジックを提供する **GroupFilter** インスタンスを作成して、次に `AddSubFilter()` メソッドを使用して、**GroupFilter** ロジックに従って組み合わせるサブフィルタ・オブジェクトを 1 つ以上割り当てます。このようにして複数の既存フィルタを組み合わせ、NLP クエリに提供するソースを選択できます。

最も簡単な GroupFilter ロジックでは、単一フィルタの反転が返ります。以下の例では、`RandFilt` がソースの 33% を選択します。GroupFilter は Negated ブーリアン演算子により AND ロジックを定義します。単一フィルタに適用されると、このロジックはフィルタで選択されないソースをすべて返します。

ObjectScript

```
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
  ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
        DO domoref.%Save()
        GOTO SetEnvironment }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { GOTO SetEnvironment }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
        QUIT}
SetEnvironment
  SET domId=domoref.Id
QueryBuild
  SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
ListerAndLoader
```

```

SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
DefineARandomFilter
SET randfilt=##class(%iKnow.Filters.RandomFilter).%New(domId,.33)
SampledSourceQueries
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The ",dname," domain contains ",numSrcD," sources",!
SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,randfilt)
WRITE "From ",numSrcD," sources randfilt sampled ",numSrcFD,!
GroupFilter
SET grpfilt=##class(%iKnow.Filters.GroupFilter).%New(domId,"AND",1)
DO grpfilt.AddSubFilter(randfilt)
SET numSrcGrp=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,grpfilt)
WRITE "From ",numSrcD," sources grpfilt sampled ",numSrcGrp

```

以下の例では、GroupFilter を使用して、2 つの他のフィルタの結果を組み合わせます（この場合、共にランダム・フィルタ）。GroupFilter ロジックは AND、かつ Negated=0 であるため、GroupFilter の結果は RandomFilter の両セットにあるソースとなります。これらのフィルタの結果はランダムであるため、GroupFilter AND の結果の数は本例の実行毎に異なる可能性があります。

ObjectScript

```

DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE { SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      GOTO SetEnvironment }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO SetEnvironment }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
SetEnvironment
SET domId=domoref.Id
QueryBuild
SET myquery="SELECT TOP 50 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
DefineTwoRandomFilters
SET randfilt1=##class(%iKnow.Filters.RandomFilter).%New(domId,.33)
SET randfilt2=##class(%iKnow.Filters.RandomFilter).%New(domId,.25)
SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
WRITE "The ",dname," domain contains ",numSrcD," sources",!
SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,randfilt1)
WRITE "From ",numSrcD," sources randfilt1 sampled ",numSrcFD,!
SET numSrcFD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,randfilt2)
WRITE "From ",numSrcD," sources randfilt2 sampled ",numSrcFD,!
GroupFilter
SET grpfilt=##class(%iKnow.Filters.GroupFilter).%New(domId,"AND",0)
DO grpfilt.AddSubFilter(randfilt1)
DO grpfilt.AddSubFilter(randfilt2)
SET numSrcGrp=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,grpfilt)
WRITE "From ",numSrcD," sources grpfilt sampled ",numSrcGrp

```

それぞれの GroupFilter に AND (\$\$\$GROUPFILTERAND) または OR (\$\$\$GROUPFILTEROR) 論理演算子を割り当てます。したがって、AND ロジックと OR ロジックの両方を使用する複合フィルタを作成するには、AND ロジックを使用した GroupFilter と OR ロジックを使用した GroupFilter を作成する必要があります。

以下の例は、ブーリアン式 "(filter1 AND !(filter2 OR filter3))" に対応しています。

ObjectScript

```
#include %IKPublic
SET domoref=##class(%iKnow.Domain).%New("MyDomain")
DO domoref.%Save()
SET domId=domoref.Id
/* . . . */
Create3Filters
/* . . . */
GroupFilters
SET group1=##class(%iKnow.Filters.GroupFilter).%New(domId,$$$GROUPFILTERAND,0)
SET group2=##class(iKnow.Filters.GroupFilter).%New(domId,$$$GROUPFILTEROR,1)
DO group1.AddSubFilter(filter1)
DO group2.AddSubFilter(filter2)
DO group2.AddSubFilter(filter3)
DO group1.AddSubFilter(group2)
```


13

テキスト・カテゴリ化

テキストのカテゴリ化により、ソース・テキストの内容に基づいて、カテゴリ・ラベルをそれらのテキストに割り当てることができます。

例えば、数多くの Aviation Event (航空イベント) テキストがあり、それらを “airplane”、“helicopter”、“glider” などの AircraftCategory ラベルで分類する場合を想定します。これらのサンプル・テキストの中で、どの NLP エンティティがカテゴリ・ラベルと強く結び付いているかを判断することで、テキスト分類モデルを作成できます。カテゴリが割り当てられていない今後の Aviation Event テキストに、このモデルを適用できます。

以下のように、適切なカテゴリを定義することは、テキストのカテゴリ化にとって欠かせない準備作業となります。

- ・ 各ソースは 1 つのカテゴリのみに割り当てることができます。1 つのソースは 1 つのカテゴリへの割り当てとなり、複数のソースに 1 つのカテゴリが割り当てられることはありません。各ソースは、定義されたカテゴリのいずれかと一義的に対応する必要があります。
- ・ カテゴリ値 (ラベル) の数は固定となります。さらなるカテゴリ・ラベルをテキスト分類モデルに追加することはできないので、将来的ソースはすべて当初のカテゴリ・ラベルのいずれかにて割り当てができるようにする必要があります。
- ・ カテゴリ値 (ラベル) の数は少なくする必要があります。カテゴリの設計において、おおよそ同数のソースが各カテゴリ値に割り当てられるようにする必要があります。

13.1 テキスト・カテゴリ化の実装

NLP では、テキスト分類モデルの構築について、以下の 2 つの手法をサポートしています。

- ・ 分析: カテゴリ・ラベルを持つ既存テキストのセットを分析して、テキスト内のいずれのエンティティがそれらの各カテゴリのメンバシップにおいて最も強力な指示子となるかを判断します。これには、カテゴリ化済みのテキストを代表的するサンプルが必要になります。
- ・ 規則基準: 既存テキストのセットについての NLP エンティティ・クエリを実施します。例えば、[上位の TFIDF または BM25 エンティティ](#)を決定します。`%AddCategory()` を使用して、カテゴリを定義します。テキスト内における価値の高いエンティティの存在について、規則 (ブーリアン・テスト) を策定して、特定のカテゴリを特定の規則で関連付けます。これらの規則を適用することで、あるカテゴリ内におけるテキストのメンバシップを一括して決めることになり、ブーリアン・テストの最高点数によってカテゴリが決定します。これにおいては、カテゴリ化済みのテキスト・セットが必要ありません。

以下の説明を、テキスト分類モデルの構築に対する分析手法に適用します。分析手法では、Naive Bayes の統計的分析やユーザ定義の決定規則など、あらゆる分析手法を使用することができます。

テキスト・カテゴリ化を実施するには、最初にテキスト分類子 (テキスト分類モデル) を作成する必要があります。このモデルは、カテゴリ・ラベル割り当て済みソース・テキストのトレーニング・セットに基づきます。それらのトレーニング・セット・テキストの内容を分析することにより、NLP はいずれの NLP エンティティがいずれのカテゴリと強く結び付くか判断します。これらの NLP テキスト・エンティティをカテゴリと統計的に関連付けたテキスト分類子を構築して、なおかつそのテストを行います。正確なテキスト分類子を得れば、それを新しいソース・テキストで使用して、カテゴリ・ラベルを割り当てることができます。

一般的に、カテゴリはメタデータ・フィールドにて指定されます。各テキストは単一のカテゴリ・ラベルに関連付けられます。各カテゴリをトレーニング・セット内の数多くのテキストにより適度に表現して、各種カテゴリ・ラベルの数をソース・テキストの数に比べて低く抑える必要があります。

NLP のテキスト分類は、ソース・テキスト内の NLP エンティティ (単語ではない) から開始します。分析においては、ソース・テキスト内エンティティの頻度だけでなく、エンティティのコンテキスト (エンティティが否定されているかどうかなど)、およびより大きなテキスト・ユニット (CRC や文など) におけるエンティティの出現も使用できます。NLP 意味分析の全領域を使用することにより、テキスト分類では高精度で価値の高いテキスト分類を実現できます。

分析的なテキスト分類は以下の 3 つのアクティビティで構成されます。

- ・ [テキスト分類子の構築](#)。これには、1 つのテキスト・セット (トレーニング・セット) が必要となり、それぞれのテキストにはカテゴリ・ラベルが割り当てられています。この手順では、それらのテキスト内に存在し、かつ差別化に役立つ可能性のある用語のセット (エンティティ) を選択します。また、トレーニング・セット・テキストにおけるそれらの用語の存在と関連カテゴリ・ラベルとがどのように相関しているかを判断する ClassificationMethod (アルゴリズム) を選択します。
- ・ [テキスト分類子をテストして](#)、その適合性を判断します。これには、別のテキスト・セット (テスト・セット) が必要となり、それぞれのテキストにはカテゴリ・ラベルが割り当てられています。このテスト情報に基づき、構築手順を再検討して、用語の追加や削除を行うことができます。これにより、テキスト分類子モデルの精度が反復的に改善することになります。
- ・ [テキスト分類子を使用して](#)、カテゴリ割り当てのないテキストを分類します。

13.1.1 実装インタフェース

テキスト分類モデルは、以下 2 つの方法のいずれかにより実装することができます。

- ・ InterSystems IRIS® Data Platform の管理ポータルにある [ユーザ・インタフェース \(UI\) ツール](#) を使用します。管理ポータルの [Analytics] オプションから、[Text Analytics] オプションを選択してから、[テキスト・カテゴリ化] を選択します。これにより、2 つのオプション [モデル・ビルダ] および [モデル・テスト] が表示されます。
- ・ [プログラムにより](#)、%iKnow.Classification パッケージ (%iKnow.Classification.Builder、%iKnow.Classification.Optimizer、および %iKnow.Classification.Classifier) が使用されます。

13.1.2 テキスト分類モデルの管理

テキスト分類子は、どのようにトレーニング、最適化、生成、テストされているかに関係なく、InterSystems IRIS によってクラス定義として格納されます。これらは、InterSystems IRIS 環境にあるその他の ObjectScript クラスと同様に管理できます。

管理ポータルには、テキスト分類モデルでテキスト分類子を容易に管理できる [\[クラス\] ページ](#) が用意されています。このページを使用すると、以下のことができます。

- ・ [テキスト分類子を削除する](#)
- ・ 他の InterSystems IRIS 環境から [テキスト分類子をインポート](#) する
- ・ 他の InterSystems IRIS 環境へ [テキスト分類子をエクスポートする準備を整える](#)

13.2 トレーニング・セットとテスト・セットの確立

いずれのインタフェースを使用するかに関係なく、テキスト分類子を構築する前には、関連付けられたカテゴリ・ラベル付きのデータ・ソースのグループをドメインにロードする必要があります。これらのソースを使用して、テキスト分類子のトレーニングとテストを行います。

注釈 関連付けられたカテゴリ・ラベル付きのソースの既存グループを必要としない規則基準テキスト分類子を作成することも可能です。ただし、この章の例においては、トレーニング・セットとテスト・セットのソースを使用することが必要となります。

ロードしたそれらのソースを(最低)2つのソース・グループに分割できるようにする必要があります。それらは、ソースのトレーニング・セットおよびソースのテスト・セットとなります。トレーニング・セットを使うことによって、いずれのエンティティが特定のカテゴリにとって適した指示子となるかを定めます。また、テスト・セット(または複数テスト・セット)を使うことによって、このカテゴリ・ラベル予測割り当てがトレーニング・セット以外のソースに対して意味を成すかどうかを判断します。これにより、特定ソース・グループに対して用語の“重複適合”を防ぎます。トレーニング・セットは2つのセットの内でソースの約70%を占める大きい方として、残り30%をテスト・セットとすることをお勧めします。

SQLソースのトレーニング・セットとテスト・セットへの分割化についての一般的方法の1つは、ソースのフィールドをメタデータ・フィールドとして使用することです。<(より小さい) および >(より大きい) の演算子を AddField() に指定すれば、そのフィールドの値に対するブーリアン・テストを実行して、ソースを2つのグループに分けることができます。このソース分割はできるだけランダムとする必要があり、通常は SQL RowID をメタデータ・フィールドとして使用することで、これを実現します。

管理ポータルの **[テキスト・カテゴリ化]** **[モデル・ビルダ]** は、メタデータ・フィールドの値を使用して、ソースのグループをトレーニング・セットとテスト・セットに分割するように設計されています。

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、[“サンプル・プログラムに関するメモ”](#)を参照してください。

以下の例では、SQL RowID をメタデータ・フィールドとして確立することで、ロードしたソースをトレーニング・セットとテスト・セットに分割するために使用できるようにしています。

ObjectScript

```
SET myquery="SELECT ID,SkyConditionCeiling,Type,NarrativeFull FROM Aviation.Event"
SET idfld="ID"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull") // text data field
SET metaflds=$LB("SkyConditionCeiling","ID")
DO ##class(%iKnow.Queries.MetadataAPI).AddField(domId,"SkyConditionCeiling") // categories field
DO ##class(%iKnow.Queries.MetadataAPI).AddField(domId,"ID",$LB("=", "<=", ">")) // set divider field
```

また、NLP ソース ID 値を使用すれば、任意のタイプのロードしたソースをグループに分割することができます。`%iKnow.Filters.SourceIdFilter` クラスの使用により、ソースのグループをトレーニング・セットとテスト・セットに分けることが可能です。以下の例では、ソース ID に対してモジュロ除算を使用することで、ロードしたソースの 2/3 を `tTrainingSet` に配置して、さらに残りを `tTestSet` に配置しています。

ObjectScript

```

FilterBySrcId
SET numsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,numsrc)
SET j=1
SET filtlist=""
WHILE $DATA(result(j)) {
    SET intId = $LISTGET(result(j),1)
    IF intId#3 > 0 {SET filtlist=filtlist_"_"intId }
    SET j=j+1
}
SET tTrainingSet=##class(%iKnow.Filters.SourceIdFilter).%New(domId,filtlist)
SET tTestSet = ##class(%iKnow.Filters.GroupFilter).%New(domId, "AND", 1) // NOT filter
DO tTestSet.AddSubFilter(tTrainingSet)
DisplaySourceCounts
SET trainsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,tTrainingSet)
WRITE "The training set contains ",trainsrc," sources",!
SET testsrc = ##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId,tTestSet)
WRITE "The test set contains ",testsrc," sources",!

```

%iKnow.Filters.RandomFilter は、ソースのグループを分割するための別手法となります。ただし、%iKnow.Filters.RandomFilter を呼び出すごとに、結果のトレーニング・セットは異なるソースで構成されます。

13.3 プログラムによるテキスト分類子の構築

テキスト分類子を構築するには、%iKnow.Classification.Builder オブジェクトを使用します。以下の説明を、テキスト分類子の構築のための分析手法に適用します。

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

13.3.1 テキスト分類子の作成

テキスト分類子を作成するには、最初にビルダ・オブジェクトをインスタンス化して、これにトレーニング・セットのドメイン名と oref を指定します。さらに、テキスト分類子が使用することになる ClassificationMethod アルゴリズムを構成します。この最も使用が簡単なアルゴリズムは Naive Bayes の定理に基づいています。Naive Bayes では、トレーニング・セットの各カテゴリに対して個々のエンティティの確率を組み合わせ、各カテゴリに属する新しいテキストの全体的確率を計算します。

ObjectScript

```

SET tBuilder = ##class(%iKnow.Classification.IKnowBuilder).%New("mydomain",tTrainingSet)
SET tBuilder.ClassificationMethod="naiveBayes"

```

さらに、テキスト分類子が使用することになるカテゴリを指定します。ソースがカテゴリ・ラベルをメタデータ・フィールドとして指定している場合、%LoadMetadataCategories() メソッドを 1 回呼び出すことができます。カテゴリ値またはカテゴリの数さえも指定する必要はありません。以下の例では、Aviation.Aircraft の AircraftCategory メタデータ・フィールドをカテゴリ・フィールドとして使用して、各レコードを 1 つのカテゴリ (“Airplane”、“Helicopter”、“Glider”、“Balloon” など) に割り当てています。

ObjectScript

```

SET myquery="SELECT TOP 100 E.ID,A.AircraftCategory,E.Type,E.NarrativeFull "_
"FROM Aviation.Aircraft AS A,Aviation.Event AS E "_
"WHERE A.Event=E.ID"
SET idfld="ID"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
SET metaflds=$LB("AircraftCategory")
SET mstat=##class(%iKnow.Queries.MetadataAPI).AddField(domId,"AircraftCategory")
IF mstat=1 {
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds,metaflds) }
.
.
.
WRITE tBuilder.%LoadMetadataCategories("AircraftCategory")

```

レコードの大半のパーセンテージ (>80%) が “Airplane” に割り当てられている一方で、他のほとんどのラベルには割り当てられたレコードがほんの一握りしかないのが、これは有益なカテゴリ・フィールドとなります (ただし、理想的ではない)。理想的には、各カテゴリ・ラベルがテキスト数とおおよそ等しく対応する必要があります。各カテゴリがトレーニング・セットの最低 10% を示している限り、ほとんどの分類メソッドは正しく機能します。1 つのカテゴリ・ラベルは複数のソース・テキストと関連付ける必要があります。したがって、潜在的なカテゴリ値をごく少数のテキストと組み合わせて、“miscellaneous” などのカテゴリ・ラベルで、包括的なカテゴリとしておくに役に立つ場合があります。

13.3.2 用語ディクショナリの生成

カテゴリを確立したら、テキスト分類子が各テキスト内を検索して、いずれのカテゴリ・ラベルを割り当てるか決めるために使用する用語を選択します。用語は個別に割り当てを行うか、または %PopulateTerms() を使用して、一定のメトリックに従いテキスト内にある複数用語を加えるかのいずれかとすることができます。

%PopulateTerms() を使用すれば、テキスト内での頻度に基づき、いくつもの用語を自動的に指定できます。既定では、Naive Bayes アルゴリズム (ほとんどがカテゴリごとの確率の差別化) を使用して、用語が選択されます。

ObjectScript

```
SET stat=tBuilder.%PopulateTerms(50)
```

Naive Bayes 以外のメトリックの実装は、サブクラスによって提供することができます。%PopulateTerms() を使用すれば、[BM25](#) または [TFIDF アルゴリズム](#) を使って、トレーニング・セット・ドキュメントの上位 X 個の用語を指定できます。

一般的には、%PopulateTerms() および %AddEntity() メソッドの組み合わせを使用して、目的の用語セットを作成することになります。

個々の用語を指定して、テキスト分類子に含めるには、以下のように %AddEntity()、%AddCRC()、および %AddCooccurrence() メソッドを使用します。

%AddEntity() では、エンティティを単一用語として加えたり、エンティティを配列やリストとして指定することで複数のエンティティを単一の複合用語として加えることができます。NLP はこれらエンティティのカウントおよびスコアを集計して、用語の同義語やグループ変異形を取得できるようにします。

ObjectScript

```

DO tBuilder.%AddEntity("hang glider")
DO tBuilder.%AddEntity("fixed wing aircraft","explicit","partialCount")
SET tData(1)="helicopter",tData(2)="helicopters",tData(3)="twin-rotor helicopter"
DO tBuilder.%AddEntity(.tData)

```

%AddEntity() は、否定および部分一致の取り扱い方法を必要に応じて指定できます。これは、前の例における 2 番目の %AddEntity() にて示しています。

%AddCRC() では、CRC を単一用語として加えることができます。テキスト分類はソース・テキストの一致頻度によって異なるので、CRC をテキスト分類子用語として役立てることは一般的ではありません。ただし、特定カテゴリに対する強力

な指示子となる、きわめて特殊な順序のエンティティ (CRC) がある場合、CRC の追加は合理的となる可能性があります。

%AddCooccurrence() によって、単一用語として同一文内に 2 つの指定エンティティの出現を追加できます (順不同)。オプションにより、否定および部分一致の取り扱い方法を指定することもできます。

ObjectScript

```
WRITE tBuilder.%AddCooccurrence($LISTBUILD("landed","helicopter pad"))
```

これらの用語は特定カテゴリに関連付けされないことに注意してください。Builder は自動的にこれらの用語を含む各テキストが各カテゴリとどの程度関連しているか計算します。

13.3.3 分類オプティマイザの実行

テキスト分類子の開発時には、試行錯誤によって用語を追加/削除する必要はありません。%iKnow.Classification.Optimizer クラスのメソッドを使用すれば、予測精度に大きく影響する該当エンティティを含めることができます。

1. オプティマイザ・オブジェクトを作成して、そのビルダ・プロパティを使用することにより、関連付けのための %iKnow.Classification.Builder オブジェクトを指定します。オプションにより、ScoreMetric プロパティを設定して、パフォーマンスの測定方法を指定します (既定は MacroFMeasure)。

ObjectScript

```
SET tOpt = ##class(%iKnow.Classification.Optimizer).%New(domId,tBuilder)
SET tOpt.ScoreMetric="MicroPrecision"
```

2. 配列から (LoadTermsArray() を使用)、または SQL クエリを使用すること (LoadTermsSQL() を使用) のいずれかによって、多数の候補用語を含めます。
3. Optimize() メソッドを実行します。これにより、用語の ScoreMetric 値に基づいて、自動的に用語の追加と削除が行われます。Optimize() では、潜在的に価値の高い用語の追加を指定回数実施して、それらの影響度を計算してから、価値の低い用語を削除します。

13.3.4 テキスト分類子の生成

カテゴリと用語を特定したら、テキスト分類子クラスを生成します。このテキスト分類子クラスには、ソースにある用語に基づいて、最適なカテゴリ・ラベルを特定するためのコードが組み込まれます。以下のように、テキスト分類子のクラス名を指定します。

ObjectScript

```
WRITE tBuilder.%CreateClassifierClass("User.MyClassifier")
```

このメソッドによって実行される操作は、指定した ClassificationMethod によって異なります。Naive Bayes では、実際のカテゴリについても把握されている各ソース・テキストの各用語に対する一致スコア/カウントを含んだマトリックスが最初に作成されます。これにより、割り当てられたカテゴリを指定用語がどの程度予測するものであるかについてのモデルを構築します。

ここで使用する例においては、カテゴリが AircraftCategory メタデータ・フィールド値より取得されたものとなります。各用語は各ソースと相関して、カテゴリの決定において、その用語の予測性がどの程度であるか判断します。例えば、用語 “helipad” の存在は AircraftCategory=helicopter を伴うソースを強く予測するものとなっています。用語 “engine” はいくつかのカテゴリ (airplane や helicopter ですが、glider や balloon ではない) を示しているので、単一カテゴリの予測としては弱くなります。ただし、このタイプの用語を含むことにより、一定カテゴリの排除には役立つことがあります。用語 “passenger” はいずれのカテゴリにとっても貧弱な予測に過ぎないので、テキスト分類子モデルには不向きな用語であ

る可能性が高くなります。`%AddEntity()` および `%RemoveTerm()` を使用すれば、カテゴリ決定の貢献度に基づいて、用語ディクショナリに適合できます。

13.4 テキスト分類子のテスト

テキスト分類子モデルがドキュメントのトレーニング・セットに適合することにより、用語ディクショナリ用語セットが正確にカテゴリを判断します。ここで、ドキュメントの別個セットにてそのモデルをテストして、トレーニング・セットのドキュメント以外においても正確であるかどうか判断する必要があります。このためには、テスト・セットのドキュメントを使用します。トレーニング・セットと同様に、これらのドキュメントにも定義されたカテゴリ・ラベルがあります。

`%TestClassifier()` メソッドを使用すれば、単一の精度値を返すことができます。この精度は、試験したレコードの合計数にて除算された適正予測の数となります。テスト・セットのドキュメントに対してこの精度が上がれば、より良好なモデルということになります。

ObjectScript

```
WRITE tBuilder.%TestClassifier(tTestSet,.accuracy),!
WRITE "model accuracy: ", $FNUMBER(100*accuracy, "L", 2), " percent"
```

すべてのカテゴリに対して、予測精度は同一にならない可能性があります。したがって、個々のカテゴリに対して精度をテストする必要があります。

以下の例では、全体の精度と個別の誤った予測結果を返します。

ObjectScript

```
TestClassifier
WRITE tBuilder.%TestClassifier(tTestSet,.testresult,.accuracy),!
WRITE "model accuracy: ", $FNUMBER(accuracy*100, "L", 2), " percent",!
SET n=1
SET wrongs=0
WHILE $DATA(testresult(n)) {
  IF $LISTGET(testresult(n),2) '= $LISTGET(testresult(n),3) {
    SET wrongcnt=wrongcnt+1
    WRITE "WRONG: ", $LISTGET(testresult(n),1)
    WRITE " actual ", $LISTGET(testresult(n),2)
    WRITE " pred. ", $LISTGET(testresult(n),3),! }
  SET n=n+1 }
WRITE wrongcnt, " out of ", n-1,!
```

カテゴリに対する予測精度は、既知のカテゴリに対する予測の一致について可能性のある以下 4 つの結果に基づいて計算されます。

- True Positive (TP): 予測は Category X であり、実際に Category X である。
- False Positive (FP): 予測は Category X であるが、実際には別のカテゴリである。
- False Negative (FN): 予測は別のカテゴリであるが、実際には Category X である。
- True Negative (TN): 予測は別のカテゴリであり、実際に別のカテゴリである。

これらのカウントを使用して、以下の比率が生成されます。

Precision は、特定のカテゴリについて返された結果の数に対する適正結果の比率となります: $TP / (TP + FP)$ 。例えば、用語 “helipad” はカテゴリ Helicopter について高精度の比率に貢献することになり、“helipad” について触れるほぼすべてのテキストはカテゴリ Helicopter に存在します。

Recall は、特定のカテゴリについて返されるはずだった結果の数に対する適正結果の比率となります: $TP / (TP + FN)$ 。例えば、用語 “helipad” は、カテゴリ “Helicopter” についての Recall 比率を改善する可能性は高くありません。なぜならば、“helipad” について触れるこれらのテキストはほんの僅かであるためです。

Category X に対するモデルの F-measure (F1) は Precision と Recall の値を組み合わせて、この 2 つの調和平均値を導き出します。Precision の増加は Recall が減少する原因となる場合があります、またその逆の場合もあります。この 2 つの内でいずれを最大化させるかは、使用事例に応じて異なります。例えば、医療スクリーニングのアプリケーションでは、より False Positives を受け入れて、False Negative 数の最小化が望まれる場合があります。

注釈 カテゴリの値がトレーニング・セットになかった場合、テキスト分類子はテスト・セットのテキストに対して、そのカテゴリを予測することはできません。この場合、True Positive (TP) および False Positive (FP) は共にゼロとなり、指定された該当カテゴリにおいて False Negative (FN) がテキストのフル・カウントとなります。

13.4.1 テスト結果の使用

トレーニング・セットの精度とテスト・セットの精度の間に重大な不一致がある場合、用語ディクショナリはトレーニング・セットに“重複適合”されていることになります。この問題を修正するには、ビルド・プロセスに戻り、用語ディクショナリを改訂します。個々の用語を用語配列で置き換えれば、用語ディクショナリを一般化することができます。

ObjectScript

```
SET stat=tBuilder.%RemoveTerm("Bell helicopter")
SET tData(1)="Bell helicopter",tData(2)="Bell 206 helicopter",tData(3)="Bell 206A helicopter",
    tData(4)="Bell 206A-1 helicopter",tData(5)="Bell 206L helicopter",tData(6)="Bell 206L LongRanger"

SET stat=tBuilder.%AddEntity(.tData)
```

また、個々の用語を変更することで、用語ディクショナリを一般化すれば、完全一致だけでなく、部分一致("partialCount")も認められるようになります。

13.5 UI を使用したテキスト分類子の構築

InterSystems IRIS 管理ポータルを使用すれば、テキスト分類子を構築できます。管理ポータルの [Analytics] オプションから、[Text Analytics] オプションを選択してから、[テキスト・カテゴリ化] を選択します。これにより、2 つのオプション [モデル・ビルダ] および [モデル・テスト] が表示されます。

すべての NLP ドメインは、特定のネームスペース内に存在します。したがって、使用するネームスペースを指定する必要があります。ネームスペースは、使用する前に、[Analytics 対応](#) しておく必要があります。対応ネームスペースを選択すると、[Text Analytics] オプションが表示されます。

注釈 NLP 処理に対して、%SYS ネームスペースを使用することはできません。%SYS ネームスペースにいる間は、管理ポータルの [Analytics] オプションが機能しません (灰色で表示)。したがって、いずれかの管理ポータル・インタフェース・ページの上部にある [切り替え] オプションをクリックして、使用する既存のネームスペースを指定する必要があります。

このインタフェースでは、既存のテキスト分類子を開くか、または新規のテキスト分類子を構築するかのいずれかが可能です。新規のテキスト分類子を構築するには、データ・ソースを含む定義済みドメインが必要となります。データ・ソースはカテゴリ・フィールドを含む必要があります。

13.5.1 UI に対するデータ・セットの定義

新規のテキスト分類子を作成するには、ドメインを作成して、トレーニング・セットとテスト・セットとして使用可能なデータ・ソースによりそのドメインを生成している必要があります。通常、これらのソースのデータは以下の指定が必要になります。

- ・ テキスト分類子により分析されるテキストを含む 1 つ以上のデータ・フィールド。

- ・ テキスト分類子により使用されるカテゴリを含むメタデータ・フィールド。データ・ソースには、可能性のある各カテゴリ値に対して最低 1 つのソースが必要となります。
- ・ データ・ソースのトレーニング・セットとテスト・セットへの分割に使用可能な値を含むメタデータ・フィールド。このフィールドはデータの実際の内容とつながりを持つ必要がないので、ソースのランダム分割が可能になります。例えば、ソース ID 番号または日付や時間の値などがそれに当たります。多くの場合、< (より小さい) および > (より大きい) の演算子を指定して、セットへのソース分割を可能にする必要があります。

以下は、データ・ソース・フィールド定義の例です。

ObjectScript

```
SET myquery="SELECT TOP 200 E.ID,A.AircraftCategory,E.Type,E.NarrativeFull "_
"FROM Aviation.Aircraft AS A,Aviation.Event AS E "_
"WHERE A.Event=E.ID"
SET idfld="ID"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
SET metaflds=$LB("AircraftCategory","ID")
DO ##class(%iKnow.Queries.MetadataAPI).AddField(domId,"AircraftCategory")
DO ##class(%iKnow.Queries.MetadataAPI).AddField(domId,"ID",$LB("=", "<=", ">"))
```

13.5.2 テキスト分類子の構築

新規のテキスト分類子を作成するには、**[新規]** ボタンを選択します。これにより、**[新規テキスト分類子の作成]** ウィンドウが表示されます。テキスト分類子の新規 **[クラス名]** を作成します。既存ドメインのドロップダウン・リストから、NLP の **[ドメイン]** を選択します。そのドメインに対して定義されたメタデータ・フィールドのドロップダウン・リストから、カテゴリ・ラベルを含む **[カテゴリ・フィールド]** を選択します。**[トレーニング・セット]** については、ドロップダウン・リストからメタデータ・フィールドを選択し、さらにドロップダウン・リストから演算子を選択して、値を指定します。例えば、EventDate <= 2004 となります。**[テスト・セット]** については、同じメタデータ・フィールドおよび補完的な演算子と値を選択します。例えば、EventDate > 2004 となります。また、SQL クエリで **[SQL]** を使用して、**[トレーニング・セット]** および **[テスト・セット]** を指定することができます。

[用語の生成] については、ドロップダウン・リストから派生用語のメソッドを選択して、派生用語の上位数を指定します。例えば、Naive Bayes (NB) に対しては、**[NB 差別化による上位 n 個の用語]** となります。その後、**[作成]** をクリックします。

これにより、以下の 3 つのパネルのスクリーンが表示されます。

右側のパネルに **[モデルのプロパティ]** が表示されます。通常は、これらの値を変更しません。データ・ソース・ドメイン名をクリックすれば、前の手順で指定した **[カテゴリ・フィールド]**、**[トレーニング・セット]**、または **[テスト・セット]** を変更することができます。ボタン・バーの **[ギア]** アイコンをクリックすれば、追加の高度制御がいくつか表示されます。

中央パネル (**[選択された用語]**) では、モデルの一部として選択済みの用語のツリー表示が現れます。左側のパネル (**[用語の追加]**) では、モデルに用語を追加することができます。

13.5.2.1 用語選択

以下に、用語ディクショナリにエンティティを追加するための最も一般的な方法を示します。

- ・ **[エンティティ]** タブ: 表示されるボックスに部分文字列を入力します。その部分文字列を含むすべてのエンティティが頻度と分散カウントを伴ってリストされます。
- ・ **[上位]** タブ: ドロップダウン・リストから **[メトリック]** (**BM25** または **TFIDF**) を選択します。そのメトリックに従った上位エンティティが算出スコアを伴ってリストされます。

チェック・ボックスを使用して、個々のエンティティを用語として選択することができます。またはリストの下部にスクロールして、リスト化されたエンティティの現在ページに対して **[すべて選択]** をクリックすることもできます。必要に応じて、他のページに進むこともできます。エンティティを選択したら、**[用語の追加]** リストの上部までスクロールで戻り、**[追加]** ボタン

をクリックします。これにより、選択したエンティティが中央パネルの **[選択された用語]** リストに追加されます。(重複用語を選択した場合、情報メッセージが表示されることがあります。)

13.5.2.2 高度用語選択 (オプション)

[追加] ボタンにより、複合用語を作成することができます。複合用語では、相似した複数のエンティティはすべて、同一の単一用語に解決されます。複合用語を定義することは、テキスト分類子のさらなる堅牢化と一般化する上において、およびトレーニング・セットの特定エンティティに対する“適合性”を低下する上において役立ちます。また、1 つの用語を別の用語の上にドラッグ・アンド・ドロップすることで、**[選択された用語]** リストに複合用語を作成することもできます。

既定では、選択した複数単語エンティティはすべて、同時出現として用語ディクショナリに追加されます。エンティティ内の単語はすべて、テキストで指定順序で出現して一致条件を満たす必要があります。ただし、代わりに個々のエンティティを部分一致エンティティとして指定することもできます。複数単語エンティティを追加する場合、**[ギア]** アイコンをクリックして、**[カウント]** ドロップダウン・ボックスを表示します。部分一致カウントを選択します。通常の複数単語エンティティを部分一致用語として定義することは、テキスト分類子のさらなる堅牢化と一般化する上において、およびトレーニング・セットの特定エンティティに対する“適合性”を低下する上においてに役立ちます。

[CRC] タブまたは **[同時出現]** タブを使用して、これらのエンティティを用語ディクショナリに追加することができます。表示されるボックスにエンティティを入力してから、**[Enter]** を押します。**[用語の追加]** リストに上位の CRC と同時出現が降順で表示されます。(同時出現とは、同一文における 2 つのコンセプト・エンティティの出現 (順不同) のことです。)通常、ほとんどの CRC と同時出現はきわめて特殊であるため、用語ディクショナリには追加しません。ただし、最も一般的な CRC または同時出現を用語ディクショナリに追加する場合もあります。

中央パネルの **[選択された用語]** リストから用語を削除することができます。これには、個々の用語 (または複合用語内のエンティティ) をクリックしてから、**[削除]** ボタンをクリックします。

13.5.2.3 保存

[モデルのプロパティ] および **[選択された用語]** リストが完成したら、**[保存]** ボタン (または **[名前を付けて保存]** ボタン) をクリックします。これにより、テキスト分類子が構築されます。

注釈 テキスト分類子は、InterSystems IRIS によって ObjectScript クラスとして保存されます。管理ポータル [の \[クラス\] ページ](#) でテキスト分類子の削除、インポート、およびエクスポートを実行できます。

13.5.3 テキスト分類子の最適化

テキスト分類子の保存後には、オブティマイザを使用して、自動的に用語ディクショナリを最適化することができます。オブティマイザは追加候補用語のリストを取得して、各用語のテストを行い、テキスト分類子の精度に最高の影響度を備えた用語を加えます。

[最適化] ボタンをクリックします。これにより、**[用語選択の最適化]** ポップアップ・ウィンドウが表示されます。ドロップダウン・リストから、**[関連メトリック]** (**BM25**、**TFIDF**、または優位性) を選択します。該当のメトリックを使用し、候補用語の数を指定してから、**[ロード]** をクリックします。右側のパネルに **[テスト用の候補用語]** が一覧表示されます。**[次へ]** をクリックします。

これにより、既定値で **[設定]** パネルが表示されます。これらの既定値を受け入れて、**[開始]** をクリックできます。これにより、オブティマイザが実行され、用語の追加と削除が行われます。最適化プロセスが完了したら、**[用語選択の最適化]** ポップアップ・ウィンドウを閉じることができます。中央パネルの **[選択された用語]** リストが変化していることに注意してください。**[保存]** ボタンをクリックして、用語ディクショナリへのこれらの追加を保存します。

オブティマイザは異なる設定にて複数回実行することができます。各最適化の後には、テキスト分類子をテストすることができます。これは、以下のセクションで説明します。

13.5.4 データのテスト・セットに対するテキスト分類子のテスト

[テスト] ボタンをクリックします。これにより、新しいブラウザ・タブに [モデル・テスト] が表示されます。[モデル・テスト] により、テスト・データに対してテキスト分類子をテストすることができます。テスト後、[モデル・ビルダ] ブラウザ・タブに戻り、手動で [用語の追加] を使用するか [オプティマイザ](#) を実行 (あるいは再実行) して、用語の追加や削除ができます。

[モデル・テスト] では、以下の 2 通りのテキスト分類子テスト方法が用意されています。

- ・ [ドメイン] タブ: [ドメイン]、[カテゴリ・フィールド]、および [テスト・フィルタ] は [モデル・ビルダ] より値を取得する必要があります。[モデル・テスト] にて、[実行] ボタンをクリックします。これにより、全体のテスト結果および各カテゴリの詳細テスト結果が表示されます。
- ・ [SQL] タブ: SQL クエリを指定すれば、データ・ソースの SQL セクションのテスト・データを提供することができます。以下の例に示すように、_Text と _Category の列エイリアスを使用して、ソース・テキストとメタデータ・カテゴリの列を指定します。

SQL

```
SELECT E.ID,A.AircraftCategory AS _Category,E.Type,E.NarrativeFull AS _Text
FROM Aviation.Aircraft AS A,Aviation.Event AS E WHERE A.Event=E.ID
```

次に、[テスト] をクリックします。これにより、テキスト分類子によって決定したカテゴリと実際のカテゴリ値が比較されます。ここでは、全体のテスト結果および各カテゴリの詳細が表示されます。

13.5.5 未分類データに対するテキスト分類子のテスト

テキスト文字列についてテキスト分類子を使用すれば、その文字列がどの程度カテゴリを導き出すかテストすることができます。[テスト] ボタンを選択します。これにより、[テキスト入力] ウィンドウが表示されます。テキスト文字列を指定してから、[分類] を押します。

- ・ [テキスト] タブでは、用語ディクショナリ用語をハイライトした形でテキストが表示されます。
- ・ [カテゴリ] タブでは、各カテゴリのスコア・バーが表示されます (緑は「正」、茶は「誤」を表す)。
- ・ [トレース情報] タブでは、入力テキストにある各用語に対する確率バーが表示されます。カテゴリのドロップダウン・リストに対してウェイトを使用すれば、すべてのカテゴリ (既定) または個々のカテゴリの各用語に対して、確率を求めることができます。

Th

13.6 テキスト分類子の使用

精度の高いテキスト分類子を構築したら、カテゴリ・ラベルが未割り当てのソース・テキストに適用してみます。`%iKnow.Classification.Classifier` のメソッドを使用すれば、テキスト分類子を使って、任意単位のテキストのカテゴリを予測することができます。

テキスト分類子を作成するには、実行したいテキスト分類子を表す `%iKnow.Classification.Classifier` のサブクラスを最初にインスタンス化する必要があります。作成後は、テキスト分類子が完全に移植可能になり、トレーニング・セットとテスト・セットのデータを含むドメインで、このテキスト分類子クラスを独立して使用できます。

- ・ NLP ソース・テキストに対して、`%Categorize()` を使用します。分類するソース・テキストが NLP ドメインでインデックス化されている場合、`%Categorize()` を使用すれば、そのカテゴリと照合できます。これにより、スコアによる降順にて、各カテゴリについての一致スコアが返ります。

ObjectScript

```
SET tClassifier = ##class(User.MyClassifier).%New("iKnow","MyDomain")
WRITE tClassifier.%Categorize(.categories,srcId)
ZWRITE categories
```

- ・ 文字列として指定したテキストに対して、%CategorizeText() を使用します。分類するソース・テキストが文字列である場合、%CategorizeText() を使用すれば、入力文字列をそのカテゴリと照合することができます。これにより、スコアによる降順にて、各カテゴリについての一致スコアが返ります。

ObjectScript

```
SET tClassifier = ##class(User.MyClassifier).%New()
WRITE tClassifier.%CategorizeText(.categories,inputstring)
ZWRITE categories
```

ZWRITE では、以下のような一致スコア・データを返します。

```
categories=4
categories(1)=$lb("AIRPLANE",.4314703807485703701)
categories(2)=$lb("HELICOPTER",.04128622469233822948)
categories(3)=$lb("GLIDER",.0228365968611826442)
categories(4)=$lb("GYROCRAFT",.005880588058805880587)
```

SQL においては、以下のように、メソッド・ストアド・プロシージャを使用して、テキスト文字列に対してテキスト分類子を実行することができます。

```
SELECT User.MyClassifier_sys_CategorizeSQL('input string') AS CategoryLabel
```

これにより、カテゴリ・ラベルが返ります。

以下の埋め込み SQL の例では、Sample.MyTC テキスト分類子を使用して、Aviation.Event の最初の 25 個のレコードについてカテゴリを判定します。

ObjectScript

```
FOR i=1:1:25 {
    SET rec=i
    &sql(SELECT %ID,NarrativeFull INTO :id,:inputstring FROM Aviation.Event WHERE %ID=:rec)
    WRITE "Record ",id
    &sql(SELECT Sample.MyTC_sys_CategorizeSQL(:inputstring) INTO :CategoryLabel)
    WRITE " assigned category: ",CategoryLabel,!
}
```

テキストが分類されたら、%iKnow.Filters.SimpleMetadataFilter クラスの使用により、この分類を使用してテキストをフィルタ処理できます。詳細は、“[ユーザ定義メタデータによるフィルタ処理](#)” を参照してください。

14

優位性と近似

NLP の意味的優位性は、`%iKnow.Semantics.DominanceAPI` と `%iKnow.Semantics.ProximityAPI` の 2 つのクラスで構成されます。これらのクラスのパラメータとメソッドの詳細は、“[インターシステムズ・クラス・リファレンス](#)”を参照してください。

この章では、以下について説明します。

- ・ [意味的優位性](#)
- ・ [語義的な近似](#)

14.1 意味的優位性

意味的優位性とは、ソースに含まれる 1 つのエンティティの全体的な重要性のことです。NLP は、以下のテストを実行し、統計的な結果を取得することで、意味的優位性を判断します。

- ・ ソース内でのエンティティの出現回数 (頻度)
- ・ ソース内でのエンティティの各構成単語の出現回数
- ・ エンティティの単語数
- ・ エンティティのタイプ (概念または関係)
- ・ ソースでのエンティティの多様性
- ・ ソースでの構成単語の多様性

無関係な単語と[パス関係](#)の単語は、優位性および頻度の計算に影響しません。

注釈 日本語については、エンティティの意味的優位性の計算に、これとは別の言語固有の統計セットを使用します。“[NLP Japanese の概要](#)”を参照してください。

NLP では、NLP のインデックス作成の一部としてソースがロードされた際にこれらの値を生成します。ここでは、これらの値を組み合わせ、各エンティティの優位性スコアを生成します。優位性スコアが高くなるほど、ソース内でのエンティティの優位性が高くなります。

例えば、概念“[心臓血管外科](#)”をドキュメント内で意味的に優位にするには、統計的分析を実行することになります。ここでは、その概念がソース内に 20 回出現すると算出されます。優位概念は、上位概念と同じではありません。このソースでは、概念の“[医者](#)” (60 回)、“[外科](#)” (50 回)、“[手術室](#)” (40 回)、および“[手術の手順](#)” (30 回) のほうがはるかに一般的です。しかし、“[心臓血管外科](#)”の構成単語は、“[心臓血管](#)” (50 回) および“[外科](#)” (80 回) と概念自体の回数よりも 2 倍多く出現し、優位性の高い概念になるサポートとなります。対照的に、概念の“[手術室](#)”は 40 回出現しますが、

その構成単語は、“手術“ が 60 回、“室“ が 45 回のみで、出現回数が概念よりわずかに多いだけです。これにより、このソースでは、心臓血管の問題や外科への関心が部屋への関心よりも高いことが示されます。

NLP では、これらの出現頻度の回数が多いまたは少ないという比重は、元のエンティティ内の単語数およびそのエンティティが概念であるか、あるいは関係であるかに基づいて指定されます。(通常、関係の発生は、概念よりも回数がはるかに少なくなります。)

ただし、概念の優位性が実際に高いかどうかを決定するには、NLP でソース内の概念の合計数と比較する必要があります。ソース内の概念の 5% に“心臓血管“と“外科“という単語が含まれ、他の概念でのこれらの単語の組み合わせがソース内での組み合わせほど頻度が高くない場合は、これらの単語はソース内で頻繁に出現するだけではなく、そのソースの内容は範囲が広くないことがわかります。しかし、そのソースで、“手“、“腎臓“および“脳“という概念内に“外科“という単語がほぼ同頻度で出現し、“心臓血管“という単語が“運動“や“ダイエット“などの単語と同頻度で出現する場合は、そのソースの内容は範囲が広いと考えられます。“心臓血管外科“という概念とその構成単語が他の単語より多く出現したとしても、そのソースの内容における優位性がそれほど高くない可能性もあります。

これらの統計的な計算を実行することによって、NLP は、ソース内の優位性の高い概念、つまり最も関心を引くサブジェクトを決定できます。NLP では、外部参照コーパス(“一般的な“医療テキストにおける単語の相対頻度を記載した既存テーブルなど)を使用することなく、この分析を実行します。NLP では、実際のソース・テキストのコンテンツのみを使用して優位性を決定し、その結果、内容に関する知識を前もって持っていなくても、すべてのトピックのソースでその優位性を使用できます。

14.1.1 コンテキストにおける優位性

NLP では、ソースに含まれるエンティティの優位性(概念と関係)を計算して、それぞれに整数の値を割り当てます。ソース内で最も優位性の高い概念には、優位性の値に 1000 を割り当てます。[インデックス作成結果ツール](#)を使用すると、1 つのソースに含まれる概念に対する優位性の値をリストできます。

NLP は、ソース内の CRC の優位性を計算します。このアルゴリズムでは、CRC 内でエンティティの優位性の値を使用します。CRC 優位性の値は、別の CRC 優位性の値と比較することのみを目的としています。CRC 優位性の値は、エンティティ優位性の値と比較することを意図していません。[インデックス作成結果ツール](#)を使用すると、1 つのソースに含まれる CRC に対する優位性の値をリストできます。

NLP は、ロードされたすべてのソースにわたる概念の優位性を加重平均として計算します。この概念優位性のスコアは小数値で、最大値は 1000 です。[ドメイン・エクスプローラ](#)・ツールの[優位性の高い概念]オプションを使用すると、ロードされたすべてのソースに含まれる概念に対する優位性スコアをリストできます。また、`%iKnow.Semantics.DominanceAPI` の `GetTop()` メソッドを使用して、ロードされたすべてのソースに含まれる概念に対する優位性スコアをリストすることもできます。

14.1.2 意味的優位性の概念

意味的優位性の主要な要素を以下に示します。

- ・ プロファイル：優位性スコアを計算するために使用する要素の数です。
- ・ 標準：標準ソースとは、そのソース内で優位性が高いエンティティがそのグループのソースで優位性が高い要素と非常に類似するソースです。これは、ブレイクの逆になります。
- ・ ブレイク：ブレイク・ソースとは、そのソースで優位性が高いエンティティがそのグループのソースで優位性が高い要素と最も類似しないソースです。これは、標準の逆になります。例えば、ブレイクの報道記事は、前月からの報道記事のすべてで優位性が高いエンティティと最も類似しないことになると考えられます。
- ・ 重複：異なるソース内における、単一エンティティの出現数です。
- ・ 相関：各ソースに対する相関の割合を返す、ソース内エンティティとエンティティ・リストとの比較です。

14.1.3 意味的優位性の例

この章では、以下のような意味的優位性を持つクエリについて説明し、その例を示します。

- ・ ドメイン内の上位の概念に対する優位性スコア
- ・ 指定エンティティに対する優位性スコア

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

14.1.3.1 ドメイン内で上位の優位性スコアを持つ概念

GetTop() メソッドは、ロードされたすべてのソースについて、優位性スコアが上位の概念（または関係）を返します。GetTop() では、フィルタと [skiplist](#) がサポートされています。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
  ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { WRITE "Deleted the data from the ",dname," domain",!!
    GOTO ListerAndLoader }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
    QUIT }
ListerAndLoader
  SET domId=domoref.Id
  SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
  SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
UseLister
  SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
  IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
  SET stat=myloader.ProcessBatch()
  IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
SourceCount
  SET numSrcD=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
  WRITE "The domain contains ",numSrcD," sources",!!
DominantConcepts
  DO ##class(%iKnow.Semantics.DominanceAPI).GetTop(.profresult,domId,1,50)
  WRITE "Top Concepts in Domain by Dominance Score",!
  SET j=1
  WHILE $DATA(profresult(j),list) {
    WRITE $LISTGET(list,2)
    WRITE ":", $LISTGET(list,3),!
    SET j=j+1 }
  WRITE !,"Printed ",j-1," dominant concepts"
```

14.1.3.2 指定エンティティに対する優位性スコア

NLP は、GetDomainValue() メソッドを使用して、指定のエンティティに対する優位性の値を返します。エンティティは、エンティティ ID（一意の整数）で指定し、数値コードでエンティティ・タイプを指定します。既定のエンティティ・タイプは 0（概念）です。

単一セットの一意のエンティティ ID は概念と関係に対して使用されるため、概念が関係と同じエンティティ ID を持つことはありません。別のセットの一意のエンティティ ID が CRC に使用されるため、CRC は、概念または関係と同じエンティティ ID を持つ可能性があります。この番号付けは全体で同時に行われ、エンティティ間での接続はありません。

以下の例では、上位 12 エンティティを取得して、各エンティティの優位性スコアを判定しています。この例の結果からわかることは、上位 (最も頻繁に出現する) のエンティティが、必ずしも高優位性スコアのエンティティと対応しているわけではないという事実です。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
TopEntitiesDominanceScores
    DO ##class(%iKnow.Queries.EntityAPI).GetTop(.result,dmId,1,12)
    SET i=1
    WHILE $DATA(result(i)) {
        SET topstr=$LISTTOSTRING(result(i),"",1)
        SET topid=$PIECE(topstr,"",1)
        SET val=$PIECE(topstr,"",2)
        SET spc=25-$LENGTH(val)
        WRITE val
        WRITE $JUSTIFY("top=",spc),i
        WRITE " dominance="
        WRITE ##class(%iKnow.Semantics.DominanceAPI).GetDomainValue(dmId,topid,0),!
        SET i=i+1 }
    WRITE "Top ",i-1," entities and their dominance scores"
```

14.2 語義的な近似

語義的な近似では、文内の 2 つのエンティティ間での意味の“隔たり”を計算します。近似の整数値が高くなるほど、エンティティは近接します。

この意味の隔たりの例として、文を以下のように指定します。

“The giraffe walked with long legs to the base of the tree, then stretched his long neck up to reach the lowest leaves.”

“キリン”という概念の近似が次のように、長い脚 = 64、ベース = 42、木 = 32、長い首 = 25、最も低い葉 = 21 であるとしてます。

語義的な近似は各文内のエンティティごとに計算され、その生成された近似スコアをまとめて追加することで、ソース・テキストの全セットにおける各エンティティ全体の近似スコアを生成します。例えば、文を以下のように指定します。

"The giraffe walked with long legs to the base of the tree, then stretched his long neck up to reach the lowest leaves. Having eaten, the giraffe bent his long legs and stretched his long neck down to drink from the pool."

“キリン”という概念の近似が次のように、長い脚 = 128、長い首 = 67、ベース = 42、木 = 32、プール = 32、最も低い葉 = 21 であるとしています。

エンティティの近似は交換可能で、つまり、エンティティ 1 とエンティティ 2 の近似は、エンティティ 2 とエンティティ 1 の近似と同じです。NLP では、エンティティ自体の語義的な近似は計算されません。例えば、“The boy told a boy about another boy.” という文では近似スコアは生成されませんが、“The boy told a younger boy about another small boy.” という文では、boy = 64、small boy = 42 という近似スコアが生成されます。同じエンティティが文に複数回出現する場合は、近似スコアが付加されます。例えば、“The girl told the boy about another boy.” という文の概念 “girl” の近似が、boy = 106 の場合は、2 つの近似スコアである 64 と 42 の合計ということになります。

14.2.1 日本語の語義的な近似

日本語の NLP 意味分析には、エンティティ・ベクトルを作成するアルゴリズムが使用されます。エンティティ・ベクトルとは、文内の、事前定義されている論理的なシーケンスに従うエンティティの順序です。NLP は、日本語の文をエンティティ・ベクトルに変換するとき、通常は、エンティティの順序を並べ替えます。日本語に対する語義的な近似には、原文のエンティティの順序ではなく、エンティティ・ベクトルのエンティティの順序が使用されます。

14.2.2 近似の例

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の例では、GetProfile() メソッドを使用して、ドメイン内のすべてのソースにある文の他の概念に対する “student pilot” という概念の近似を返します。GetProfile() では、[フィルタ](#)と [skiplist](#) がサポートされています。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
    SET dname="mydomain"
    IF (##class(%iKnow.Domain).NameIndexExists(dname))
    { WRITE "The ",dname," domain already exists",!
      SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
      GOTO DeleteOldData }
    ELSE
    { WRITE "The ",dname," domain does not exist",!
      SET domoref=##class(%iKnow.Domain).%New(dname)
      DO domoref.%Save()
      WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
      GOTO ListerAndLoader }
DeleteOldData
    SET stat=domoref.DropData()
    IF stat { WRITE "Deleted the data from the ",dname," domain",!!
      GOTO ListerAndLoader }
    ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
    SET domId=domoref.Id
    SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
    SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
    SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
    SET idfld="UniqueVal"
    SET grpfld="Type"
    SET dataflds=$LB("NarrativeFull")
UseLister
    SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
    IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
    SET stat=myloader.ProcessBatch()
    IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
```

```

ProximityForEntity
  SET entity="student pilot"
  DO ##class(%iKnow.Semantics.ProximityAPI).GetProfile(.eresult,domId,entity,1,20)
  SET k=1
  WHILE $DATA(eresult(k)) {
    SET item=$LISTTOSTRING(eresult(k))
    WRITE $PIECE(item,"",1)," ^ "
    WRITE $PIECE(item,"",2)," ^ "
    WRITE $PIECE(item,"",3),!
    SET k=k+1 }
  WRITE !,"all done"

```

以下の例では、GetProfileBySourceId() メソッドを使用して、各ソースについて、指定されたエンティティに対する最大の近似を持つ概念をリストします。各概念は、エンティティ ID、値、および近似スコアでリストされます。

ObjectScript

```

#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { WRITE "The ",dname," domain already exists",!
    SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
  ELSE
  { WRITE "The ",dname," domain does not exist",!
    SET domoref=##class(%iKnow.Domain).%New(dname)
    DO domoref.%Save()
    WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
    GOTO ListerAndLoader }
DeleteOldData
  SET stat=domoref.DropData()
  IF stat { WRITE "Deleted the data from the ",dname," domain",!!
    GOTO ListerAndLoader }
  ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
    QUIT}
ListerAndLoader
  SET domId=domoref.Id
  SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
  SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
  SET myquery="SELECT Top 25 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
  SET idfld="UniqueVal"
  SET grpfld="Type"
  SET dataflds=$LB("NarrativeFull")
UseLister
  SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
  IF stat '= 1 {WRITE "The lister failed: ", $System.Status.DisplayError(stat) QUIT }
UseLoader
  SET stat=myloader.ProcessBatch()
  IF stat '= 1 {WRITE "The loader failed: ", $System.Status.DisplayError(stat) QUIT }
SourceCountQuery
  SET totsrc=##class(%iKnow.Queries.SourceAPI).GetCountByDomain(domId)
GetEntityID
  SET entId=##class(%iKnow.Queries.EntityAPI).GetId(domId,"student pilot")
QueryBySource
  DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId,1,totsrc)
  SET j=1,k=1
  WHILE $DATA(result(j),srclist) {
    SET src = $LISTGET(srclist)
    WRITE !,"Source id: ",src,!
    SET entity="student pilot"
    DO ##class(%iKnow.Semantics.ProximityAPI).GetProfileBySourceId(.srcresult,domId,entId,src,1,totsrc)

    WHILE $DATA(srcresult(k)) {
      SET item=$LISTTOSTRING(srcresult(k))
      WRITE $PIECE(item,"",1)," ^ "
      WRITE $PIECE(item,"",2)," ^ "
      WRITE $PIECE(item,"",3),!
      SET k=k+1 }
    SET k=1
    SET j=j+1 }
  WRITE !,"Printed all ",j-1," sources"

```


15

カスタム・メトリック

ユーザがエンティティ、CRC、CC、パス、文、およびソースについて独自のカスタム・メトリックを追加できるように、NLPをカスタマイズできます。これらのメトリックは、ドメイン内のすべてのソースについて生成することも、選択したソースについて生成することもできます。その後、これらのカスタム・メトリック値によるクエリを実行できます。

注釈 カスタム・メトリック・ソフトウェアはテクノロジー・プレビューです。テクノロジー・プレビューの目的は、インターシステムズが既存および将来のアプリケーションの有効性を高めるために役立つと確信している新しいソフトウェア機能を紹介して利用可能にするための方法を提供することです。

ここに示す機能は、ユーザがいつでも使用できますが、機能や設計の面でまだ完全ではありません。これらの機能を利用するユーザは、以下の点を理解する必要があります。

- ・ インターシステムズは、今後の更新に関して下位互換性を保証しません。
- ・ ユーザは、これらの機能を配置済みのアプリケーションに組み込むことができますが、その前にインターシステムズに問い合わせ、最善の対策を決定する必要があります。
- ・ これらの機能をアプリケーションに導入するユーザは、最終リリース・バージョンへのアップグレードを確約する必要があります。

ソフトウェアにこれらの項目を組み込むユーザには、自身の経験に関するフィードバックを提供することを強くお勧めします。

15.1 カスタム・メトリックの実装

カスタム・メトリックを実装するには、以下の手順を実行します。

1. メトリックを定義する：1 つまたは複数のメトリックを定義するには、`%iKnow.Metrics.MetricDefinition` のサブクラスを実装し、1 つまたは複数のカスタム・メトリックのプロパティを指定する “Metrics” という名前の XData ブロックを追加します。
2. メトリックの計算を定義する：これらのメトリックの値を計算するには、`%iKnow.Metrics.MetricBuilder` サブクラスに計算を実装します。このサブクラスでは、`Calculate***Metrics()` メソッドを 1 つまたは複数実装して、対応するターゲットおよびタイプのカスタム・メトリックの計算をサポートする必要があります。例えば、`CalculateEntUniMetrics()` は、ドメイン全体に適用可能なメトリックのターゲット・エンティティを計算します。
3. メトリック定義を登録する：メトリック定義を特定のドメインに関連付けるには、`%iKnow.Metrics.MetricDefinition` クラスの `Register()` メソッドを呼び出して、実装した `%iKnow.Metrics.MetricBuilder` サブクラスを “ビルダ・クラス” として定義に登録します。これにより、すべてのメトリック、それらのプロパティ、およびそれらのターゲットが “Metrics”

XData ブロックの定義に従って登録されます。メトリック定義を登録すると、メトリック値を格納するのに必要なデータ構造が構成されます。

%iKnow.DomainDefinition の [サブクラスとしてドメインが作成され](#)、メトリックがそのドメインに対して定義された場合、Register() メソッドを呼び出して、メトリック定義を登録する必要はありません。

4. メトリック値を構築する：%iKnow.Metrics.MetricBuilder の実装を使ってメトリック値を構築するには、ビルダ・クラスの Build() または BuildAll() メソッドを呼び出します。これらのメソッドは、適用可能なすべてのメトリックを構築し、実際のメトリックの計算を Calculate***Metrics() の実装に転送します。フィルタを指定して、カスタム・メトリックの構築を定義されたソースのサブセットだけに限定できます。

%iKnow.Metrics.MetricBuilder サブクラスで SUPPORTSMULTIPROCESS パラメータ (既定値 = 1) をオーバーライドしない限り、Build() メソッドは作業を複数のプロセスに分割します。各プロセスは、別個の MetricBuilder オブジェクトをインスタンス化し、それらのビルダ・オブジェクトを使用してエントリのバッチを次々に処理します。ソースのインデックスが作成されたターゲット・タイプの場合、これらはソースのバッチであり、すべてのターゲット要素が同じビルダ・プロセスによって処理されます。

5. 構築プロセスを最適化する：%iKnow.Metrics.MetricBuilder には、いくつかのコールバック・ユーティリティ・メソッドが用意されています。例えば、OnBeforeBuildMetrics()、OnBeforeBuildMetricsBatch()、OnBeforeBuildMetricsForSource()、およびこれらに対応する OnAfter***() メソッドなどがあります。これらのメソッドを使用すると、バッチ全体の値を事前に計算して、その値を個々の Calculate***Metrics() メソッドで選択できるため、パフォーマンスを最適化するのに役立ちます。個々の Calculate***Metrics() メソッドは、出力パラメータ pValues を使って値を返します。pValues は、pValues("MyMetric") = [MyValue] のようにメトリック名でインデックス指定して生成されます (1 つの MetricBuilder クラスが 1 つのパスで複数のメトリックをサポートして計算できるため)。
6. 結果を取得する：%iKnow.Metrics.MetricAPI には、特定のドメインに登録されたカスタム・メトリックを操作するために使用できるメソッドがいくつか用意されています。これらのメソッドを使用して、カスタム・メトリックの結果を返すことができます。

Calculate***Metrics() メソッドが特定のターゲット要素の値を返さないときは、既存のメトリック値が (あれば) 保持されます。値として "" を返すと、既存の値が上書き (消去) されます。

使用例は、Aviation.Metrics.Builder クラスと Aviation.Metrics.Definition クラスを参照してください。

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、["サンプル・プログラムに関するメモ"](#)を参照してください。

15.2 タイプおよびターゲット

- ・ メトリックは、メトリック値がドメイン全体のコンテキスト内で適用されるか、ある特定のソースまたはメタデータ・グループだけに適用されるかを指定するタイプを持っています。使用可能な値は、\$\$\$IKMTRTYPEDOMAIN、\$\$\$IKMTRTYPESOURCE、および \$\$\$IKMTRTYPEGROUP です。

\$\$\$IKMTRTYPEGROUP メトリックは、特定のメタデータ・フィールドで定義されたソース・グループのコンテキストで有効です。このタイプのメトリックのメトリック・ビルダ・クラスでは、Calculate***Metrics() の実装の一部として CurrentFieldName および CurrentFieldValue プロパティの値を使用できます。

- ・ 1 つのメトリックは、カスタム・メトリックが適用される要素であるターゲットを 1 つまたは複数持っています。使用可能な値は、\$\$\$IKMTRTYPEDOMAIN、\$\$\$IKMTRTYPESOURCE、および \$\$\$IKMTRTYPEGROUP です。ターゲットは、どの Calculate***Metrics() メソッドを指定するかによって指定されます。例えば、CalculateEntUniMetrics() はターゲット \$\$\$IKMTRTYPEDOMAIN に適用されます。適用方法は、タイプによって異なります。

15.3 メトリックのコピー

現在のネームスペース内のドメイン間でメトリックをコピーできます。

- ・ `%iKnow.Utils.CopyUtils` クラスの `CopyMetrics()` メソッドを使用すると、ドメイン内のすべてのメトリック定義を別のドメインにコピーできます。このメソッドでは、オプションで、1 つのドメインから別のドメインにメトリック値をコピーすることもできます。
- ・ `%iKnow.Utils.CopyUtils` クラスの `CopyMetric()` メソッドを使用すると、ドメイン内の 1 つのメトリック定義を別のドメインにコピーできます。このメソッドはメトリック値をコピーしません。
- ・ `%iKnow.Utils.CopyUtils` クラスの `CopyDomain()` メソッドを使用すると、ドメイン・コピー処理の一部としてすべてのメトリック定義をコピーできます。このメソッドでは、オプションで、ドメイン・コピー処理の一部としてメトリック値をコピーすることもできます。

16

スマート・マッチング：ディクショナリの作成

スマート・マッチングとは、NLP インデックス作成プロセスの結果を、ディクショナリ、分類法、またはオントロジといった形で所有している外部の知識と組み合わせることです。こうしたインデックス作成結果により、どの単語をまとめて概念や関係を形成するかが特定されるため、一致の品質を判断することが可能になり、NLP マッチングが“スマート”になります。例えば、NLP は、ディクショナリ用語である“flu”に一致する語が、実際にインデックス付きテキスト・ソースで“flu”という概念を示しているのか、それとも“bird flu”という概念を示しているのかを識別できます。後者は部分一致と呼ばれますが、この場合の一致は、インデックス付けされたテキスト・ソースのエンティティにディクショナリ用語が正確に対応する完全一致の場合とは異なる扱いが必要なこと（または異なる扱いが可能なこと）は明らかです。

スマート・マッチングを実行するには、ディクショナリを作成または取得する必要があります。ディクショナリを作成する場合は、マッチングに使用したい項目や用語を入力する必要があります。ディクショナリを生成したら、そのコンテンツを使用して[マッチング処理を実行](#)できます。

注釈 現時点では、ディクショナリの定義は日本語をサポートしていません。

この章では、以下について説明します。

- ・ [ディクショナリの概念構造と用語](#)
- ・ [ディクショナリの作成、および項目と用語のディクショナリへの入力](#)
- ・ [ドメイン依存ディクショナリとドメイン非依存ディクショナリ](#)
- ・ [ディクショナリ形式の用語の作成](#)
- ・ [ディクショナリのリスト](#)
- ・ [ディクショナリのコピー](#)
- ・ [ディクショナリ構造の拡張](#)

16.1 ディクショナリ構造とマッチングの概要

NLP ディクショナリを生成するには、まず項目を作成し、次に1つ以上の用語をその項目に関連付けます。ディクショナリは通常、複数項目から構成され、それぞれの項目は複数の用語に関連付けられています。項目は、ソース・テキストの多数のエンティティの関連タグとなる単語または語句です。ソース・テキストのエンティティが一致すると判別されると、その項目でタグ付けされます。例えば、項目“ship”は、“ship”、“boat”、“sail”、“oars”などの関連タグです。

このマッチングを実行するには、ディクショナリの各項目にマッチングさせる用語を入力します。用語は、1つのエンティティ（例：“motor boat”）または語句や文（例：“boats are rowed with oars or paddles”）です。NLP は、ソース・テキストに使用されているのと同じ言語モデルを使用して、ディクショナリ内の各用語のインデックスを作成します。次に NLP

は、各用語をソース・テキストの同じコンテンツ・ユニットとマッチングさせます (コンセプト用語はソース・テキストのコンセプトに対して、CRC 用語はソース・テキストの CRC に対してマッチングさせます)。NLP は、用語とソース・テキスト・ユニットが一致することを識別すると、関連付けられたディクショナリ項目でそのソース・テキストにタグ付けします。このマッチングは同一でないことがよくありますが、用語とソース・テキストが一致すると見なしてタグ付けしていいかどうかを判断するために、NLP はスコアリング・アルゴリズムを使用することが必要とされます。

NLP ディクショナリ機能は、現在のドメインに対して語幹解析が有効になっている場合、[語幹解析](#)をサポートします。つまり、単一のディクショナリ用語をソース・テキスト内の同一単語の別の形式と照合できます。

16.1.1 用語

ディクショナリは、互いに論理的関連性のある異なる用語をグループ化するための方法です。ディクショナリは、例えば、都市、ICD10 コード、またはフランス産ワインとすることができます。ディクショナリは、マッチング API 内で使用される集約のレベルで、現実のどのレベルのグループ化をディクショナリに対応させるかは、それぞれの使用事例によって異なります。上位のレベル (例：“すべての ICD10 コード”) を使用すると、パフォーマンスが高まって使用するディスク容量が減りますが、低位のレベル (例：“全 ICD10 カテゴリの個々のカテゴリ”) を使用すると、より詳細にグループ化された結果になる場合があります。各ディクショナリには名前と説明があります。

ディクショナリ項目は、ディクショナリ内で一意に識別することが可能な項目です。ディクショナリ項目の例としては、都市、ICD10 の個々のコード、または個々のシャトーが挙げられます。各ディクショナリには、通常、多数のディクショナリ項目が含まれます (少数の項目を持つ小さなディクショナリを多数作成すると、パフォーマンスが低下します)。ディクショナリ項目は URI を持ちます。これはドメイン内で一意であることが必要で、外部識別子およびオプションの説明として使用できます。この URI は、後でマッチング結果を解釈するためのルールを作成するときに使用できます。

ディクショナリ用語とは、テキストのどこかに出現する可能性のある文字列で、それが属するディクショナリ項目を表します。例えば、“Antwerp”、“Anvers”、“Antwerpen” は、Antwerp という都市を表す同じディクショナリ項目に関連付けられた異なる用語の可能性があります。ディクショナリ用語はフリー・テキスト文字列で、文字列ベースのマッチングを行う際の実際のマッチングはこれに基づいて行われます。ディクショナリ項目が表すものの異なる綴り、翻訳、または同義語である可能性があります。これらの文字列はエンジンを通して渡されて、単なる 1 つのエントリティを超えるものを含む場合は、単独の概念の境界を越えてマッチングできるように、より複雑な構造に自動的に変換されます (CRC またはパス)。ディクショナリ用語をエンジンで処理することが必要な場合は、言語も関連付けられているはずです。

NLP エンジンを通して新しいディクショナリ用語を渡すことでこれを処理すると、用語内で識別された異なるエントリティを表すために 1 つ以上のディクショナリ要素が生成されます。例えば、ディクショナリ用語の “failure of the liver” は、“failure”、“of”、および “liver” の 3 つの要素に変換され、“the” は無関係として破棄されます。これらの要素は自動的に生成、管理され、ある種の出力のみに関係するため、あまり気にする必要はありません。

特別な形式の日付や数字など、形式が設定された文字列を識別するには、ディクショナリ形式を使用してそれらを指定します。その後、完全な用語を表すものとして、または複雑な用語の 1 つの要素として、これらをディクショナリ用語に含めることができます。形式とは、日付形式といった意味のある文字パターンです。例えば、“nn/nn/nnnn” および “nnnn-nn-nn” という形式を Date という項目と関連付けることができます。NLP は、ソース・テキストにこれらの形式が出現すると、Date 項目でタグ付けします。

注釈 NLP には、日付、時刻、期間、および測定のみ一般的な表現にフラグを設定する [意味的属性](#) が用意されています。ディクショナリ形式を定義する前に、お使いの言語でこれらの属性を使用できるかどうか、およびお使いの言語におけるこれらの属性の特定性を確認してください。

16.2 ディクショナリの作成

ディクショナリを定義するには、このセクションで説明されているディクショナリの定義および生成用の `%iKnow.Matching.DictionaryAPI` クラス・メソッドを使用します。ドメイン専用のディクショナリを定義したり、ドメイン非依存で現在のネームスペースの任意のドメインで使用できるディクショナリを定義したりできます。

%iKnow.Matching.DictionaryAPI には、新しいディクショナリを作成し、それに項目、用語、および形式を割り当てるための多数のメソッドが用意されています。

- ・ CreateDictionary() は、NLP ディクショナリを作成するために使用します。
1 番目の引数は、ドメイン ID を整数に指定します。ドメインにディクショナリを割り当てるには、そのドメイン ID を正の整数として指定します。ディクショナリをドメイン非依存として定義するには、そのドメイン ID として 0 を指定します。2 番目の引数には、意味のあるディクショナリ名を指定できます。残りの引数は省略可能です。3 番目の引数にはディクショナリの説明を入力し、4 番目には言語を指定 (既定は英語)、そして 5 番目には [カスタム・マッチング・プロファイル](#) を入力します。CreateDictionary() は、一意の整数である dictId を返します。このディクショナリ ID は、以降のスマート・マッチング・メソッドで使用されます。指定された名前のディクショナリが既に存在している場合、CreateDictionary() は -1 を返します。
- ・ CreateDictionaryItem() は、ディクショナリ内で項目を作成するために使用します。dictId を指定すると、CreateDictionaryItem() によって一意の整数である dictItemId が返されます。
- ・ CreateDictionaryTerm() は、用語を既存の項目と関連付けるために使用します。dictItemId を指定すると、CreateDictionaryTerm() によって一意の整数である dictTermId が返されます。
- ・ CreateDictionaryItemAndTerm() は、特定の場合作に使用できるショートカットです。これを使用して項目を作成し、用語と項目の値が同じ場合は、その項目に関連付けられる用語を作成できます。例えば、“flu” という項目には、いくつかの用語 (“influenza”、“le grippe”、“bird flu”、“H1N1” など) を関連付けられますが、CreateDictionaryItemAndTerm() を使用すると、項目 “flu” を作成し、関連付けられる用語の “flu” をそれに割り当てることができます。もちろん、2 つのメソッド呼び出し (CreateDictionaryItem() および CreateDictionaryTerm()) を使用して、同じ処理を実行することも可能です。
- ・ CreateDictionaryTermFormat() は、形式から構成される用語を既存の項目と関連付けるために使用します。dictItemId を指定すると、CreateDictionaryTermFormat() によって一意の整数である dictTermId が返されます。

16.2.1 ディクショナリおよびドメイン

作成する各ディクショナリは、ドメイン固有にするか、ドメイン非依存にして現在のネームスペースの任意のドメインで使えるようにすることができます。

- ・ ドメイン固有のディクショナリは、CreateDictionary() メソッドで domainId を指定することでドメインに割り当てられます。ディクショナリの項目、用語、および形式に対して同じ domainId を指定します。このメソッドは、連続する正の整数として dictId を返します。このディクショナリを使用するマッチング・メソッドは、この dictId によってこれを参照します。
- ・ ドメイン非依存のディクショナリはドメインに割り当てられません。代わりに CreateDictionary() メソッドで 0 の domainId を指定します。ディクショナリの項目、用語、および形式に対しても domainId を 0 に指定します。このメソッドは、連続する正の整数として dictId を返します。このディクショナリを使用するマッチング・メソッドは、負の dictId によってこれを参照します。例えば、dictId 8 で識別されるディクショナリは dictId 値 -8 で参照されます。

ドメイン非依存のディクショナリの使用は、[語幹解析](#)の結果に重要な役割を果たします。ドメインが語幹抽出済みで構成されているためにディクショナリ用語とソース・テキストが一致する場合、通常用語のドメイン固有ディクショナリを作成すると、NLP は自動的にディクショナリ用語の語幹解析を実行します。ドメイン非依存のディクショナリを作成すると、ディクショナリ用語の語幹変換は実行されません。ユーザは通常の (語幹抽出されていない) 用語のディクショナリまたは語幹抽出済み用語のディクショナリのいずれかを作成できます。ドメイン非依存の通常用語のディクショナリでは、語幹抽出済みドメインに対する照合はできません。ドメイン非依存の語幹抽出済み用語のディクショナリでは、語幹抽出されていないドメインに対する照合はできません。

いくつかのドメインすべてが同じ dictId 値のドメイン固有のディクショナリを持つことができるとちょうど同じように、ドメイン固有のディクショナリとドメイン非依存のディクショナリの両方が同じ整数の dictId 値を持つことができます。ディクショナリ・マッチング処理は、ドメイン固有のディクショナリ (正の整数 ID で指定) とドメイン非依存のディクショナリ (負の整数 ID で指定) の任意の組み合わせを使用できます。

マッチング結果を返すマッチング API のクエリは、マッチングがドメイン非依存ディクショナリのエントリに対応する場合、(dictId, itemId, および termId に) 負の識別子を返します。すべてのクエリは、ドメイン固有のディクショナリのマッチングおよびドメイン非依存のディクショナリのマッチングの結果の組み合わせを返します。ただし、dictId パラメータで指定された値に応じて、ドメイン固有のディクショナリまたはドメイン非依存のディクショナリのいずれかの結果のみを返す、GetDictionaryMatches() と GetDictionaryMatchesById() は例外です。既定は、ドメイン固有のディクショナリのマッチングです。

16.2.2 ディクショナリ作成例

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の例では、“AviationTerms”という名前のディクショナリを作成し、2つの項目とそれらに関連付けられる用語を入力します。このディクショナリは特定のドメインに割り当てられます。

ObjectScript

```
SET domId=##class(%iKnow.Domain).GetOrCreateId("mydomain")
/* ... */
CreateDictionary
SET dictname="AviationTerms"
SET dictdesc="A dictionary of aviation terms"
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname,dictdesc)
IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
             GOTO ResetForNextTime }
ELSE {WRITE "created a dictionary ",dictId,!}
PopulateDictionaryItem1
SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(domId,dictId,
                                "aircraft",domId_dictId_"aircraft")
SET termId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
                                "airplane")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
                                "helicopter")
PopulateDictionaryItem2
SET itemId2=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItemAndTerm(domId,dictId,
                                "weather",domId_dictId_"weather")
SET i2termId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
                                "meteorological information")
SET i2term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
                                "visibility")
SET i2term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
                                "winds")
DisplayDictionary
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryItemsAndTerms(.result,domId,dictId)
SET i=1
WHILE $DATA(result(i)) {
    WRITE $LISTTOSTRING(result(i),"",1),!
    SET i=i+1 }
WRITE "End of items in dictionary ",dictId,!
/* ... */
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "deleted dictionary ",dictId,!}
ELSE {WRITE "DropDictionary error ",$System.Status.DisplayError(stat) }
```

以下の例では、前の例と同じディクショナリが作成されますが、このディクショナリは現在のネームスペース内の任意のドメインで使用できます。

ObjectScript

```
CreateDictionary
SET dictname="AviationTerms"
SET dictdesc="A dictionary of aviation terms"
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(0,dictname,dictdesc)
IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
             GOTO ResetForNextTime }
ELSE {WRITE "created a dictionary ",dictId,!}
PopulateDictionaryItem1
SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(0,dictId,
                                "aircraft",0_dictId_"aircraft")
SET termId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId,
```

```

    "airplane")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId,
    "helicopter")
PopulateDictionaryItem2
SET item2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItemAndTerm(0,dictId,
    "weather",0_dictId_"weather")
SET i2term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,item2Id,
    "meteorological information")
SET i2term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,item2Id,
    "visibility")
SET i2term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,item2Id,
    "winds")
DisplayDictionary
SET domId=##class(%iKnow.Domain).GetOrCreateId("mydomain")
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryItemsAndTerms(.result,0,dictId)
SET i=1
WHILE $DATA(result(i)) {
    WRITE $LISTTOSTRING(result(i),"",1),!
    SET i=i+1
}
WRITE "End of items in dictionary ",dictId,!
/* ... */
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(0,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(0,dictId)
IF stat {WRITE "deleted dictionary ",dictId,!}
ELSE {WRITE "DropDictionary error ",$System.Status.DisplayError(stat) }

```

16.2.3 形式用語の定義

%iKnow.Matching.Formats パッケージは、以下の 3 つのシンプルな形式クラスを提供します。

- ・ **%iKnow.Matching.Formats.SimpleDateFormat** : さまざまな形式の日付と時刻をマッチングします。サポートされている日付/時刻形式の一覧は、“ObjectScript リファレンス”の“\$ZDATETIMEH”関数を参照してください。日付および時刻の識別に関する NLP のサポートについては、“[時間、期間、および頻度](#)”を参照してください。
- ・ **%iKnow.Matching.Formats.SimplePrefixFormat** : 指定の接頭語文字列で始まるエンティティをマッチングします。
- ・ **%iKnow.Matching.Formats.SimpleSuffixFormat** : 指定の接尾語文字列で終わるエンティティをマッチングします。

必要に応じて、追加の形式クラスを作成できます。

以下の例では **%iKnow.Matching.Formats.SimpleSuffixFormat** を使用しています。ここでは最初に、speed という 1 つの項目を含むディクショナリを定義します。“speed”項目には、“excessive speed”および接尾語形式用語“mph”（マイル / 時）の 2 つの用語が含まれます。この接尾語形式は、接尾語“mph”で終わるあらゆるエンティティ（例：“65mph”）と一致します。

ObjectScript

```

SET domId=##class(%iKnow.Domain).GetOrCreateId("mydomain")
/* ... */
CreateDictionary
SET dictname="Traffic"
SET dictdesc="A dictionary of traffic enforcement terms"
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname,dictdesc)
IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
    GOTO ResetForNextTime }
ELSE {WRITE "created a dictionary ",dictId,!}
CreatedictionaryItemAndTerms
SET
item1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(domId,dictId,"speed",domId_dictId_"speed")

SET term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,item1Id,
    "excessive speed")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTermFormat(domId,
    item1Id,"%iKnow.Matching.Formats.SimpleSuffixFormat",$LB("mph",0,3))
WRITE "dictionary=",dictId,!, "item=",item1Id,!, "terms=",term1Id, " ",term2Id,!!
/* ... */
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "deleted dictionary ",dictId,!}
ELSE {WRITE "DropDictionary error ",$System.Status.DisplayError(stat) }

```

16.2.4 ディクショナリ用語における複数の形式

ディクショナリ用語の一部としてディクショナリ形式を直接入力できます。これにより、1 つ以上の形式要素と文字列要素を含む複数の要素で構成されるディクショナリ用語を作成できます。

この機能を使用するには、CreateDictionaryTerm() メソッドに送信される文字列の一部としてコード化された説明を指定します。このコード化された説明は以下の形式になります。

```
@@@User.MyFormatClass@@@param1@@@param2@@@
```

この説明は、形式クラスの完全なクラス名 (%iKnow.Matching.Formats.Format の実装)、@@@ 区切り文字、および形式クラスに渡される形式パラメータの @@@ 区切りリストで構成されています。この説明全体の始めと終わりが @@@ マーカで区切られています。

形式クラスがパラメータを取得していない場合、または既定が使用されている場合は、@@@ マーカによって区切られた形式クラス名を指定します。

ディクショナリ用語の文字列にこの形式を追加する場合は、1 つのエンティティとして NLP に確実に認識させる必要があります。例えば、"was born in @@@User.MyYearFormat@@@" という用語は 1 つのエンティティとして解釈されますが、"was born in the year @@@User.MyYearFormat@@@" という用語はそうのように解釈されません。

指定された形式クラスが NLP で見つけられない場合は、@@@ の使用が意図的であると見なされ、エンティティ全体が単純な文字列要素として処理されます。

この構文を使用することで、形式に対する別の手順またはアクションが必要なくなり、ファイルまたはテーブルからのディクショナリのロードが容易になります。

16.3 ディクショナリのリストおよびコピー

%iKnow.Matching.DictionaryAPI クラスには、既存ディクショナリおよびその項目や用語をカウントまたはリストするためのメソッドが多数用意されています。

%iKnow.Utils.CopyUtils クラスには、単一もしくはすべてのディクショナリをドメイン間でコピーするためのメソッドが多数用意されています。

16.3.1 既存のディクショナリのリスト

以下の例は、ドメイン内のすべてのディクショナリをリストします。デモンストレーションのために、この例では、最初に 2 つの空のディクショナリ (1 つは既定の言語である英語、もう 1 つはフランス語) を作成します。

ObjectScript

```
SET domId=##class(%iKnow.Domain).GetOrCreateId("mydomain")
SET dictname1="Diseases",dictname2="Maladies"
SET dictdesc1="English disease terms",dictdesc2="French disease terms"
CreateFirstDictionary
SET dictId1=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname1,dictdesc1)
IF dictId1 = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname1)
    SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
    IF stat '= 1 { WRITE "DropDictionary error ", $System.Status.DisplayError(stat)
                  QUIT }
    GOTO CreateFirstDictionary }
ELSE {WRITE "created a dictionary ",dictId1,!}
CreateSecondDictionary
SET dictId2=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname2,dictdesc2,"fr")
IF dictId2 = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname2)
    SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
    IF stat '= 1 { WRITE "DropDictionary error ", $System.Status.DisplayError(stat)
```

```

        QUIT }
    GOTO CreateSecondDictionary }
ELSE {WRITE "created a dictionary ",dictId2,!}

GetDictionaries
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaries(.dicts,domId)
WRITE "get dictionaries status is:",$System.Status.DisplayError(stat),!!
SET k=1
WHILE $DATA(dicts(k)) {
    WRITE $LISTTOSTRING(dicts(k)),!
    SET k=k+1 }
WRITE "End of list of dictionaries"

```

GetDictionaries() は、各ディクショナリの ID、名前、説明、および言語をリストします。

16.3.2 ディクショナリのコピー

現在のネームスペース内で、1 つのドメインから別のドメインにディクショナリをコピーできます。

- **%iKnow.Util.CopyUtils** クラスの **CopyDictionaries()** メソッドを使用すると、ドメイン内のすべての定義済みディクショナリを別のドメインにコピーできます。既定では、このメソッドは 1 つのドメインから別のドメインにマッチング・プロファイルをコピーすることもできます。
- **%iKnow.Util.CopyUtils** クラスの **CopyDictionary()** メソッドを使用すると、ドメイン内の単一の定義済みディクショナリを別のドメインにコピーできます。既定では、このメソッドは 1 つのドメインから別のドメインにマッチング・プロファイル ID をマップすることもできます。
- **%iKnow.Util.CopyUtils** クラスの **CopyDomain()** メソッドを使用すると、ドメイン・コピー処理の一環としてディクショナリをコピーできます。

16.4 ディクショナリ構造の拡張

NLP ではマッチング API にある単純なディクショナリのみが記述されていますが、このことによって、オントロジ、分類法、またはその他のより階層型の構造など、より高度なツールの使用が制限されることはありません。マッチング API の目的は、シンプルなマッチングの仕組みを提供することで、あらゆる構造をカバーする一般的な構造をまた 1 つ追加することではありません。つまり、手持ちのオントロジや分類法の構造を単に平坦化すればいいのです。ディクショナリ項目の URI を適切に選択することにより、オントロジや分類法のコンテキスト内でマッチング結果を再構築または解釈することが可能になります。

例えば、**%iKnow.Matching.Formats.Format** インタフェースの実装による正規表現マッチングを行うクラスの実装など、マッチング API では独自のクラス実装を提供できるという点で、フォーマット・ビットは着脱可能です。

17

スマート・マッチング：ディクショナリの使用

スマート・マッチングとは、NLP インデックス作成プロセスの結果を、ディクショナリ、分類法、またはオントロジといった形で所有している外部の知識と組み合わせることです。こうしたインデックス作成結果により、どの単語をまとめて概念や関係を形成するかが特定されるため、一致の品質を判断することが可能になり、NLP マッチングが“スマート”になります。例えば、NLP は、ディクショナリ用語である“flu”に一致する語が、実際にインデックス付きテキスト・ソースで“flu”、“bird flu”、または“no flu symptoms”という概念を示しているのかを識別できます。2 番目のケースは部分一致と呼ばれますが、この場合の一致は、インデックス付けされたテキスト・ソースのエンティティにディクショナリ用語が正確に対応する完全一致の場合とは異なる扱いが必要なこと（または異なる扱いが可能なこと）は明らかです。3 番目のケースの一致では、否定付きの部分一致と認識されます。

スマート・マッチングを実行するには、ディクショナリを作成または取得する必要があります。[ディクショナリの作成と入力](#)については、前の章で説明されています。ディクショナリを生成したら、そのコンテンツを使用してマッチング処理を実行できます。

17.1 ディクショナリ・マッチングの仕組み

NLP は、ソース・テキスト内の同じレベルの構造に対して、ディクショナリの各用語をマッチングさせます（コンセプト用語はソース・テキストのコンセプトに対して、CRC 用語はソース・テキストの CRC に対してマッチングさせます）。こうして一致する語は、完全一致または部分一致になります。用語とソース・テキストが完全一致の場合、NLP ではその用語に関連付けられたディクショナリ項目でそのソース・テキストの一節にタグを付けます。用語とソース・テキストが完全一致ではない場合、NLP はそれらの一致の程度をスコアリングします。この一致スコアには、各コンポーネントのエンティティ（概念または関係）に対する一致スコアの計算が含まれ、必要な場合は、このエンティティの一致スコアを使用して、CRC、パスまたは文の一致スコアを計算します。構成された最小一致スコアに部分一致のスコアが達すると、NLP ではその用語に関連付けられたディクショナリ項目でそのソース・テキストの一節にタグを付けます（%iKnow.Matching.MatchingProfile の MinimalMatchScore プロパティを参照してください）。

17.1.1 一致スコア

NLP では、一致スコア、浮動小数点数（ディクショナリ用語間で検出されたエンティティの一致のすべてを計算）、およびソース・テキストのユニットが生成されます。エンティティの一致では、その一致スコアが 0（一致なし）および 1（完全一致）の間の範囲になります。CRC 一致またはパス一致では、その一致がこの範囲を含みますが、1 より大きくなることもあります。このスコアの計算に使用されるアルゴリズムは複雑ですが、以下の考慮事項があります。

- ・ エンティティの完全一致は、完全（同じ順番の同じ単語）または散在（異なる順番の同じ単語）があります。散在一致の一致スコアは、マッチング・プロファイルの ScatteredMatchMultiplier プロパティの値に 1（完全一致）を乗算することで確定されます。

- ・ エンティティ (概念または関係) の部分一致では、ディクショナリ用語と一致するソース・テキストの文字列の割合に基づいてパーセンテージが割り当てられます。
- ・ エンティティの部分一致で、一致する関係に割り当てられる値は、一致する概念の半分のみです。マッチング・プロファイルの RelationshipScoreMultiplier プロパティを設定して、この割合を変更できます (例えば、関係一致と概念一致を等しく評価します)。
- ・ CRC、パスまたは文のマッチング時には、エンティティの一致から一致スコアが追加されてから、ディクショナリ用語の長さによる除算、マッチング・エンティティの数による乗算、そして DisorderMultiplier プロパティ、すなわちソース・テキストのユニットとディクショナリ用語の間での不一致の程度を表す値による乗算が実行されます。
- ・ エンティティが否定の一部である場合、エンティティの一致スコアは変更できます。既定では、NegationMultiplier のプロパティ値は 1 となり、一致スコアの計算では肯定エンティティと否定されたエンティティを等価として扱うようになります。通常、この既定をお勧めします。NegationMultiplier を 0 に設定することもでき、これにより一致スコアの計算では否定されたエンティティをスキップします。ほとんどの場合、0 の値によってこれらの一致が全体的にスキップされることとなりますが、十分な否定されない一致エンティティとの複合一致により、MinimalMatchScore しきい値を超えるスコアを取得する場合は例外です。また、このプロパティを 0 ~ 1 の値に設定することもできます。これにより、否定されたエンティティのエンティティ・レベルの一致スコアが変更されるので、それらのスコアは部分一致と見なされます。例えば、値が 0.5 では、否定されたエンティティに対してエンティティ・レベルのスコアを得ることになります。

前述の式は、一致スコアを取得するための完全な数式ではありません。これは、NLP による一致スコアの計算時に使用される主な考慮事項を示すために用いられています。

NLP に用意されているマッチング・プロファイル (%iKnow.Matching.MatchingProfile) は、前述の数値プロパティなどで構成されています。NLP では、一致スコアの計算時にこのマッチング・プロファイルが使用されます。NLP では、マッチング・プロファイル用に既定のプロパティ値が用意されています。特に指定がない場合は、この既定のマッチング・プロファイルが各ディクショナリに割り当てられます。この既定のマッチング・プロファイルによって、多くのアプリケーションに対応する正確なマッチングが提供されます。[カスタム・マッチング・プロファイル](#)の作成と割り当てについてはこの章の残り部分で説明されます。

17.2 文字列のマッチング

%iKnow.Matching.MatchingAPI クラスを使用して、テキスト文字列と生成された 1 つまたは複数のディクショナリとの間でマッチングを実行できます。文字列のマッチングには、以下の 2 種類があります。

- ・ エンティティ長文字列のマッチング
- ・ 複数エンティティを含む文字列のマッチング。この文字列は 1 つ以上の文を含む場合があります。

17.2.1 エンティティ文字列のマッチング

GetDictionaryMatches() メソッドは、単一エンティティ文字列をディクショナリとマッチングさせて、一致した項目を返します。このメソッドは文字列を単一エンティティとして扱うので、きわめて固有の一致情報を得ることができます。

GetDictionaryMatches() は文字列変数を取得するので、単一エンティティ文字列をインデックス化して、ディクショナリとマッチングさせる必要はありません。

ObjectScript

```
SET domn="entitytestdomain"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
  SET domId=domo.Id
}
ELSE {
```

```

    SET domo=##class(%iKnow.Domain).%New(domn)
    DO domo.%Save()
    SET domId=domo.Id }
/* ... */
CreateDictionary
SET dictname="AviationTerms"
SET dictdesc="A dictionary of aviation terms"
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname,dictdesc)
IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
    GOTO ResetForNextTime }
ELSE {WRITE "created a dictionary ",dictId,!}
PopulateDictionary
SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(domId,dictId,
    "aircraft",domId_dictId_"aircraft")
SET term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "airplane")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "single-engine airplane")
SET term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "helicopter")
DisplayDictionary
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryItemsAndTerms(.result,domId,dictId)
SET i=1
WHILE $DATA(result(i)) {
    WRITE $LISTTOSTRING(result(i),"",1),!
    SET i=i+1 }
WRITE "End of items in dictionary ",dictId,!
DoMatching
SET mystring="A small single-engine two-person airplane cabin"
SET stat=##class(%iKnow.Matching.MatchingAPI).GetDictionaryMatches(.num,domId,mystring,$LB(dictId))
IF stat'=1 {WRITE "get matches status is:",$System.Status.DisplayError(stat),!
    QUIT }
WRITE "The string is: ",mystring,!
WRITE "The matches are:",!
SET j=1
WHILE $DATA(num(j)) {
    WRITE "match number ",j," is ",$LISTTOSTRING(num(j)),!
    SET j=j+1 }
WRITE "End of match items for dictionary ",dictId,!
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "deleted dictionary ",dictId,! }

```

GetDictionaryMatches() は一致単語のビット・マップを提供することで、文字列内の各単語とディクショナリ用語との一致を表示します。例えば、00101 はディクショナリ用語 “single-engine airplane” と文字列 “A small single-engine two-person airplane / cabin” との一致を示します。このビット・マップは、マッチングが完了すると停止するため、単語 “cabin” のビットはありません。この例において、単語 “single-engine” と “airplane” はディクショナリ用語と文字列で同じ順序であるため、散在ブーリアンは 0 です。

各ディクショナリのマッチングでは、一致要素のリストが返されます。前の例におけるマッチングでは、以下のように一致要素のリストが返されます (ID 番号は各自で異なる場合があります)。

位置	意味	例における値
1	ディクショナリ ID	6
2	項目 ID	5
3	ディクショナリ項目 URI (ドメイン ID + ディクショナリ ID + 項目の値)	26aircraft
4	用語 ID	13
5	用語の値	single-engine airplane
6	要素 ID (用語 ID と同じ)	13
7	一致タイプ (term、format、または unknown)	用語
8	一致スコア	.333333
9	一致単語のビット	00101
10	散在ブーリアン	0
11	形式出力	null

17.2.2 文文字列のマッチング

GetMatchesBySource() メソッドは、複数エンティティ文字列をディクショナリとマッチングさせて、一致した項目を返します。(通常、このような文字列は文の長さ (またはそれ以上) となります。)最初に文字列をインデックス化してから、ディクショナリとマッチングさせる必要があります。以下の例は、AviationTerms ディクショナリに対して文字列をマッチングさせます。

ObjectScript

```

DomainCreateOrOpen
SET dname="onestringdomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  GOTO LoadString }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO LoadString }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
  QUIT}
LoadString
SET domId=domoref.Id
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
SET ^mystring="A single-engine airplane reported poor visibility and strong gusty winds. "_
  "No winds were predicted by local airport ground personnel."
DO myloader.BufferSource("ref",^mystring)
DO myloader.ProcessBuffer()
GetExtId
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,dmId,1,20)
SET i=1
WHILE $DATA(result(i)) {
  SET extId = $LISTGET(result(i),2)
  SET i=i+1 }
CreateDictionary
SET dictname="AviationTerms"
SET dictdesc="A dictionary of aviation terms"
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,dictname,dictdesc)
IF dictId=-1 {WRITE "Dictionary ",dictname," already exists",!
  GOTO ResetForNextTime }
ELSE {WRITE "created a dictionary ",dictId,!}
PopulateDictionaryItem1

```

```

SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(domId,dictId,
    "aircraft",domId_dictId_"aircraft")
SET term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "single-engine airplane")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId,
    "helicopter")
PopulateDictionaryItem2
SET itemId2=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItemAndTerm(domId,dictId,
    "weather",domId_dictId_"weather")
SET i2term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
    "strong winds")
SET i2term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
    "visibility")
SET i2term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(domId,itemId2,
    "winds")
DisplayDictionary
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryItemsAndTerms(.result,domId,dictId)
SET i=1
WHILE $DATA(result(i)) {
    WRITE $LISTTOSTRING(result(i),"",1),!
    SET i=i+1 }
WRITE "End of items in dictionary ",dictId,!
DoMatching
SET stat=##class(%iKnow.Matching.MatchingAPI).GetMatchesBySource(.num,domId,extId,$LB(dictId))
IF stat'=1 {WRITE "get matches status is:",$System.Status.DisplayError(stat),!
    QUIT }
WRITE "The string is: ",^mystring,!
WRITE "The matches are:",!
SET j=1
WHILE $DATA(num(j)) {
    WRITE "match ",j,": ditem ",$LISTGET(num(j),4),
        " dterm ",$LISTGET(num(j),5),
        " matchscore ",$LISTGET(num(j),8),
        " negated? ",$LISTGET(num(j),15),!
        SET j=j+1 }
WRITE "End of match items for dictionary ",dictId,!
ResetForNextTime
IF dictId = -1 {
    SET dictId=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryId(domId,dictname)}
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "deleted dictionary ",dictId,! }
ELSE { WRITE "DropDictionary error ",$System.Status.DisplayError(stat) }

```

以下は、文字列の“No winds”と項目“weather”のディクショナリ用語“winds”とのマッチングとなります。(この一致文字列は%List形式で返され、5,6,12,26weather,33,0,6,.5,1,0,1,1,1,1,1のようにコンマ区切りの文字列で表示されます。)これらの要素の値について、以下に説明します。

位置	意味	例における値
1	一致数	5
2	ディクショナリ ID	6
3	項目 ID	12
4	ディクショナリ項目 URI (ドメイン ID + ディクショナリ ID + 項目の値)	26weather
5	用語 ID	33
6	ターゲット・タイプ	0
7	ターゲット ID	6
8	一致スコア	.5
9	一致概念カウント	1
10	一致関係カウント	0
11	部分一致カウント	1
12	パスの最初の一致位置	1
13	パスの最後の一致位置	1

位置	意味	例における値
14	が整列	1
15	否定されたエンティティのカウント	1

17.3 ソースのマッチング

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

GetTotalItemScoresBySource() メソッドは、ソースをディクショナリとマッチングさせて、それぞれのディクショナリ項目について一致スコアを返します。以下の 2 つの例では、ドメイン固有のディクショナリを使用します。ディクショナリの定義、およびディクショナリを参照するあらゆる %iKnow.Matching.DictionaryAPI メソッドにおいて、ドメイン ID を 0 に指定します。ドメイン内でディクショナリを使用するあらゆる %iKnow.Matching.MatchingAPI メソッドにおいて、ディクショナリ ID を負の数に指定します。したがって、SET dictId=-^mydictId となります。

以下の例では、ドメイン内の全ソースを AviationTermsND ディクショナリとマッチングさせて、それぞれのディクショナリ項目について各ソースの一致スコアを返します。このプログラムを実行する前に、ディクショナリを作成する必要があります。

ObjectScript

```
CreateDictionary
SET ^mydictname="AviationTermsND"_$HOROLOG
SET dictdesc="A dictionary of aviation terms"
SET ^mydictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(0,^mydictname,dictdesc)
IF ^mydictId=-1 {WRITE "Dictionary ",^mydictname," already exists",!
    QUIT }
ELSE {WRITE "created a dictionary ",^mydictId,!}
PopulateDictionaryItem1
SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(0,^mydictId,
    "aircraft",0_^mydictId_"aircraft")
SET term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId,
    "single-engine airplane")
SET term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId,
    "helicopter")
PopulateDictionaryItem2
SET itemId2=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItemAndTerm(0,^mydictId,
    "weather",0_^mydictId_"weather")
SET i2term1Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId2,
    "strong winds")
SET i2term2Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId2,
    "visibility")
SET i2term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId2,
    "winds")
PopulateDictionaryItem3
SET itemId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryItem(0,^mydictId,
    "flight plan",0_^mydictId_"flight plan")
SET term3Id=##class(%iKnow.Matching.DictionaryAPI).CreateDictionaryTerm(0,itemId,
    "flight plan")
DisplayDictionary1
SET stat=##class(%iKnow.Matching.DictionaryAPI).GetDictionaryItemsAndTerms(.result,0,^mydictId)
WRITE "Status is: ",stat,!
SET i=1
WHILE $DATA(result(i)) {
    WRITE $LISTTOSTRING(result(i),"",1),!
    SET i=i+1 }
WRITE "End of items in dictionary ",^mydictId,!!
```

ObjectScript

```
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
    SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
    GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
```



```

        SET domoref=##class(%iKnow.Domain).%New(dname)
        DO domoref.%Save()
        WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
        GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
        GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT Top 10 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
DoMatching
SET dictId=-^mydictId
SET num=0
FOR j=1:1:10 {SET extId=##class(%iKnow.Queries.SourceAPI).GetExternalId(domId,j)
  SET mstat=##class(%iKnow.Matching.MatchingAPI).GetTotalItemScoresBySource(.mresult,domId,
    extId,$LB(dictId))
  IF mstat '=1 {WRITE "End of sources",! QUIT }
  SET k=1
  IF $DATA(mresult(k))=0 {WRITE "no dictionary matches for this source",!}
  ELSE {
    WHILE $DATA(mresult(k)) {
      WRITE $PIECE($LISTTOSTRING(mresult(k)),"",2)," "
      WRITE $PIECE($LISTTOSTRING(mresult(k)),"",4)
      WRITE " matches: ", $PIECE($LISTTOSTRING(mresult(k)),"",6)
      WRITE " score: ", $PIECE($LISTTOSTRING(mresult(k)),"",7),!
      SET k=k+1 }
    }
  SET srcname=$PIECE($PIECE(extId,":",3,4),"\\",$1(extId,"\\"))
  WRITE "End of ",srcname," match items for dictionary ",dictId,!!
}
}

```

GetMatchesBySource() メソッドは、各ソースをディクショナリとマッチングさせて、一致した項目を返します。

以下の例では、ドメイン内の全ソースをドメイン固有の AviationTermsND ディクショナリ (上記にて定義) とマッチングさせて、各ソースの一致項目を一致スコアおよび否定 (存在する場合) 付きで返します。

ObjectScript

```

DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT Top 10 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=flister.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }

```

```

DoMatching
SET dictId=-^mydictId
FOR j=1:1:10 {SET extId=##class(%iKnow.Queries.SourceAPI).GetExternalId(domId,j)
SET stat=##class(%iKnow.Matching.MatchingAPI).GetMatchesBySource(.num,domId,extId,$LB(dictId))
IF stat'=1 {WRITE "get matches status is:",$System.Status.DisplayError(stat),!
QUIT }
WRITE "The matches are:",!
SET k=1
WHILE $DATA(num(k)) {
IF $LISTGET(num(k),15)>0 {WRITE "neg. "}
WRITE "match ",k," : ditem ",$LISTGET(num(k),4),
" dterm ",$LISTGET(num(k),5),
" matchscore ",$LISTGET(num(k),8),!
SET k=k+1 }
WRITE !,"Next Source",!
}
WRITE "End of match items for dictionary ",dictId,!

```

17.4 マッチング・プロファイルの定義

ディクショナリ用語とソース・テキストのユニットが一致するかどうかは、%iKnow.Matching.MatchingProfile のマッチング・プロファイル・プロパティで決まります。8 個のプロパティがあり、これらが連携して、一致かどうかを判断します。これらのプロパティはすべて、適切な既定値を使用します。既定では、すべてのディクショナリに既定のマッチング・プロファイルが割り当てられます。

既定とは異なる値を持ったプロパティを 1 つまたは複数有するカスタム・マッチング・プロファイルを作成できます。さらに、このカスタム・マッチング・プロファイルをディクショナリに割り当てることができます。カスタム・マッチング・プロファイルで指定されないすべてのプロパティには既定の値が取得されます。カスタム・マッチング・プロファイルを作成する数には制限がありません。同じマッチング・プロファイルを複数のディクショナリに適用できます。マッチング・プロファイルは、1 つのドメインに特定するか、またはネームスペースのすべてのドメインで使用できるように定義できます。

以下の例では、3 つのカスタム・マッチング・プロファイルを作成しています。1 番目はドメイン固有のもので、オプションのマッチング・プロファイル名を指定しています。NLP では、それに正の整数の ID を割り当てます。2 番目と 3 番目はネームスペースの全ドメインに対して有効です。NLP では、それぞれに負の整数の ID を割り当てます。3 番目ではオプションのマッチング・プロファイル名を指定しているので、ドメイン ID パラメータのプレースホルダとして 0 を指定する必要があります。

ObjectScript

```

SET domn="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
SET domId=domo.Id
}
ELSE {
SET domo=##class(%iKnow.Domain).%New(domn)
DO domo.%Save()
SET domId=domo.Id }
MatchingProfile
SET domprof=##class(%iKnow.Matching.MatchingProfile).%New(domId,"mydomainCMP")
WRITE "profile Id=",domprof.ProfileId,!
WRITE "profile domain=",domprof.DomainId,!
WRITE "profile name=",domprof.Name,!
SET allprof1=##class(%iKnow.Matching.MatchingProfile).%New()
WRITE "profile Id=",allprof1.ProfileId,!
WRITE "profile domain=",allprof1.DomainId,!
WRITE "profile name=",allprof1.Name,!
SET allprof2=##class(%iKnow.Matching.MatchingProfile).%New(0,"namespaceCMP")
WRITE "profile Id=",allprof2.ProfileId,!
WRITE "profile domain=",allprof2.DomainId,!
WRITE "profile name=",allprof2.Name

```

以下の例は、カスタム・マッチング・プロファイルを定義して、ディクショナリに割り当てする方法を示しています。ここでは、カスタム・マッチング・プロファイル・インスタンス oref(ここでは、customprofile)を CreateDictionary() メソッドに指定して、既定値をオーバーライドします。

ObjectScript

```
#include %IKPublic
SET domn="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(domn))
{ SET domo=##class(%iKnow.Domain).NameIndexOpen(domn)
  SET domId=domo.Id
}
ELSE {
  SET domo=##class(%iKnow.Domain).%New(domn)
  DO domo.%Save()
  SET domId=domo.Id }
CustomizeMatchingProfile
SET customprofile=##class(%iKnow.Matching.MatchingProfile).%New()
WRITE "MinimalMatchScore initial value=",customprofile.MinimalMatchScore,!
SET customprofile.MinimalMatchScore=".4"
WRITE "MinimalMatchScore custom value=",customprofile.MinimalMatchScore,!
CreateDictionary
SET dictId=##class(%iKnow.Matching.DictionaryAPI).CreateDictionary(domId,"mydict","","en",customprofile)

WRITE "created a dictionary with ID=",dictId,!
/* . . . */
CleanUpForNextTime
SET stat=##class(%iKnow.Matching.DictionaryAPI).DropDictionary(domId,dictId)
IF stat {WRITE "Dropped the dictionary"}
ELSE {WRITE "DropDictionary error",$System.Status.DisplayError(stat)}
```

17.4.1 マッチング・プロファイル・プロパティ

ディクショナリとソース・テキストのユニットの用語が一致するかどうかは、カスタム・マッチング・プロファイルで定義したマッチング・プロファイル・プロパティの値で決定されます。このプロパティによって、各ユニットのテキストの一致スコアと、一致としてレポートされる最小一致スコアを指定するしきい値が決定されます。ディクショナリのコンテンツは使用事例に応じて異なる傾向があるため、ディクショナリのマッチング・プロファイル・プロパティを1つまたは複数カスタマイズすることが必要な場合もあります。これらのプロパティのいずれかを変更すると、レポートされる一致の数が大幅に変化する可能性があります。したがって、全体のデータセットにディクショナリをマッチングするマッチング・プロパティを登録する前に、小さなサブセットのデータによるテストでプロパティ値の変更を試しておくことが必要な場合もあります。

%iKnow.Matching.MatchingProfile のプロパティの意味については、クラス・ドキュメントで説明されています。

17.4.1.1 MinimalMatchScore

調整用の最も一般的なマッチング・プロファイル・プロパティは MinimalMatchScore です。このプロパティ値は、保存される一致の下限のしきい値です。これは、1 (保存された完全一致のみ) と 0 (保存された一致の可能性のあるものすべて) の間の小数値で設定可能で、既定値は 0.33 です。

- このプロパティの値を増加させることで、一致候補の数が多すぎる場合は、低質の一致をフィルタで除外できます。ディクショナリがかなり一般的で、多くの一般用語を含む場合は、これが適切だと思われます。これらの一般用語のいくつかに対して非常に近い一致が起こる CRC およびパスのみをレポートさせることが必要な場合もあります。
- このプロパティの値を低下させることで、一致候補の数を増やすことができます。ディクショナリが非常に特殊で、常にフラグの設定が必要な重要用語のみで構成されている場合は、これが適切だと思われます。一致が緩やかではあるが重要で、それを見逃すことがないように、技術用語のディクショナリで緩やかに一致する CRC およびパスのレポートが必要な場合もあります。

MinimalMatchScore を 0 に設定すると、一致の可能性のあるすべての結果が返されます。小さなサブセットのデータに新しいディクショナリをマッチングさせる作業を始める場合は、MinimalMatchScore を 0 に設定してから徐々に増加させ、フィルタによる低質な一致の除外を妥当な数の結果が得られるまで続ける作業も開始できます。

17.4.2 ドメインの既定マッチング・プロファイル

MAT:DefaultProfile (\$\$\$IKPMATDEFAULTPROFILE) ドメイン・パラメータを設定することで、異なるドメイン全体に既定のマッチング・プロファイルを指定できます。(接頭辞の “MAT:” は、マッチング処理に固有のドメイン・パラメータである

ことを示します。)以下の例では、ドメイン固有のカスタム・マッチング・プロファイルを定義した後に、mydomain ドメインの既定マッチング・プロファイルとしてその定義済プロファイル割り当てます。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { SET domo=##class(%iKnow.Domain).NameIndexOpen(dname) }
  ELSE
  { SET domo=##class(%iKnow.Domain).%New(dname) DO domo.%Save() }
  SET domId=domo.Id
  WRITE "domain ",dname," has Id ",domId,!!
CreateProfile
  SET domprof=##class(%iKnow.Matching.MatchingProfile).%New(domId,"mydomainCMP")
  WRITE "profile Id=",domprof.ProfileId,!
  WRITE "profile domain assignment=",domprof.DomainId,!
  WRITE "profile name=",domprof.Name,!
  WRITE "profile properties:",!
  WRITE "default MinimalMatchScore=",domprof.MinimalMatchScore,!
  SET domprof.MinimalMatchScore=.27
  WRITE "changed to MinimalMatchScore=",domprof.MinimalMatchScore,!!
  DO domprof.%Save()
MakeProfileDomainDefault
  SET str=domo.GetParameter($$$IKPMATDEFAULTPROFILE,.dpin)
  WRITE "domain ",domId," DefaultProfile before SET=",dpin,!
  SET sc=domo.SetParameter($$$IKPMATDEFAULTPROFILE,"mydomainCMP")
  IF sc=1 {
  DO domo.GetParameter($$$IKPMATDEFAULTPROFILE,.dpout)
  WRITE "domain ",domId," DefaultProfile after SET=",dpout,! }
  ELSE {WRITE "SetParameter error",! }
CleanUp
  DO ##class(%iKnow.Domain).%DeleteId(domo.Id)
  WRITE "All done"
```

以下の例では、ネームスペースのカスタム・マッチング・プロファイル (ドメイン固有ではない) を定義した後に、mydomain ドメインの既定マッチング・プロファイルとしてその定義済プロファイル割り当てます。0: が SetParameter(\$\$\$IKPMATDEFAULTPROFILE,"0:nodomainCMP") のプロファイル名の前にあることに注意してください。

ObjectScript

```
#include %IKPublic
DomainCreateOrOpen
  SET dname="mydomain"
  IF (##class(%iKnow.Domain).NameIndexExists(dname))
  { SET domo=##class(%iKnow.Domain).NameIndexOpen(dname) }
  ELSE
  { SET domo=##class(%iKnow.Domain).%New(dname) DO domo.%Save() }
  SET domId=domo.Id
  WRITE "domain ",dname," has Id ",domId,!!
CreateProfile
  SET domprof=##class(%iKnow.Matching.MatchingProfile).%New(0,"nodomainCMP")
  WRITE "profile Id=",domprof.ProfileId,!
  WRITE "profile domain assignment=",domprof.DomainId,!
  WRITE "profile name=",domprof.Name,!
  WRITE "profile properties:",!
  WRITE "default MinimalMatchScore=",domprof.MinimalMatchScore,!
  SET domprof.MinimalMatchScore=.27
  WRITE "changed to MinimalMatchScore=",domprof.MinimalMatchScore,!!
  DO domprof.%Save()
MakeProfileDomainDefault
  SET str=domo.GetParameter($$$IKPMATDEFAULTPROFILE,.dpin)
  WRITE "domain ",domId," DefaultProfile before SET=",dpin,!
  SET sc=domo.SetParameter($$$IKPMATDEFAULTPROFILE,"0:nodomainCMP")
  IF sc=1 {
  DO domo.GetParameter($$$IKPMATDEFAULTPROFILE,.dpout)
  WRITE "domain ",domId," DefaultProfile after SET=",dpout,! }
  ELSE {WRITE "SetParameter error",! }
CleanUp
  DO ##class(%iKnow.Domain).%DeleteId(domo.Id)
  WRITE "All done"
```

また、マッチング処理に影響を与える個々のドメイン・パラメータも設定できます。

17.4.2.1 1つの関係との一致

既定で、NLP マッチング・アルゴリズムでは、1 つの関係エンティティと一致するディクショナリ用語がスキップされます。例えば、ディクショナリ用語に “to” がある場合に、NLP では、“Pete goes to work” という文のエンティティ “goes to” との一致を試みません。この既定によって、ディクショナリが主に概念を含み、ターゲットにする場合のマッチング・パフォーマンスが最適化されます。

ただし、ディクショナリで意図的に 1 つの関係要素をターゲットにしている場合、[MAT:SkipRelations \(\\$\\$IKPMATSKIPRELS\)](#) ドメイン・パラメータを 0 に設定することで、[このドメイン・パラメータの既定値を変更](#)できます。これにより、エンティティのタイプとは関係なく、すべてのエンティティがすべてのディクショナリ用語に一致するようになります。

このオプションは、マッチング・プロファイル・プロパティではなく、ドメイン・パラメータとして設定されます。その理由は、このスキップ手順は、一致した用語が何かまだわからない時点で発生し、その結果、ディクショナリ専用のどのプロファイルにも適用できないからです。したがって、これはドメイン・パラメータとして実行されることで、ドメイン内のすべてのディクショナリに確実に適用されます。

17.4.2.2 その他のマッチング処理ドメイン・パラメータ

MAT:SkipRelations、MatchScoreMargin、FullMatchOnly および EntityLevelMatchOnly など、設定可能なドメイン・パラメータのいくつかは一致処理に影響を与える可能性があります。[これらのドメイン・パラメータを変更](#)する場合は、以前のドメイン・パラメータ設定を使用して一致したソースをすべて明示的に再一致させて、新しいドメイン・パラメータ値を反映させる必要があります。

ドメイン・パラメータの詳細は、このドキュメントの付録 “[ドメイン・パラメータ](#)” を参照してください。

18

ユーザ・インタフェース

NLP テクノロジは既定のユーザ・インタフェースを備えていません。この章では、NLP と共に提供されるいくつかのサンプル・ユーザ・インタフェースについて説明します。お客様の NLP の使用方法に固有のクエリ・インタフェースを開発する際は、これらに基づいて着手すると便利です。

これらのサンプル・ユーザ・インタフェースは `%iKnow.UI` パッケージにあり、Web ページとして実装されています（“インターシステムズ・クラスリファレンス”を参照）。

18.1 NLP ユーザ・インタフェースの表示方法

1. InterSystems IRIS® Data Platform ランチャーからスタジオにアクセスします。
2. [ファイル] のドロップダウン・メニューから、[ネームスペース変更] を選択します。`%SYS` ネームスペースを選択します。[OK] をクリックします。
3. [ファイル] のドロップダウン・メニューから、[開く] を選択します。
4. オープン・ウィンドウから、[システムアイテムを含む] にチェックが付いていることを確認します。その後、[%iKnow]→[UI] で目的のユーザ・インタフェースを選択します。[開く] をクリックします。ユーザ・インタフェースのソース・コードがクラス・エディタ（メイン・スタジオ・ウィンドウ）に表示されます。
5. [ブラウザで表示] アイコン（地球アイコン）をクリックします。既定のブラウザにユーザ・インタフェースが表示されます（抽象クラスは表示できません）。

詳細は、“[スタジオの使用法](#)” のドキュメントを参照してください。

18.2 抽象ポータル

これは、`%iKnow.UI` パッケージの他のすべてのポータルとページのスーパークラスです。そして、ドメイン ID の処理、「選択された」ソース ID、メタデータ・フィルタ、および NLP クエリ駆動型テーブルとグループのページングといった再使用可能な多数の素材をグループ化します。これは抽象で、それだけで実行することはできませんが、対応するペイン（`optDomainPane`、`txtTermPane`、`optSourcePane`、および `filterPane`）が使用されている限り、サブクラスまたはコンポーネント名を制限することはできません。

`%iKnow.UI.AbstractPortal` クラスには 5 つのサブクラスがあり、これらは以下の 2 つに分類されます。

- ・ ロード・ウィザード：ソース・ファイルの管理インタフェースを提供します。

- ・ クエリ・ポータル・インタフェース：ソース・データのさまざまな表示を提供して、NLP インデックス作成とディクショナリ・マッチングを表示します。

18.3 抽象ソース・ビューワ

このクラスは、AbstractPortal スーパークラスを拡張したものです。これには、ProcessInput() および DeleteCurrentVirtualSource() メソッドが含まれます。抽象であるため、それ自体で実行することはできません。

18.4 ロード・ウィザード

管理とメンテナンスのためのインタフェースで、ユーザはこれを使用してドメインおよび構成オブジェクトを簡単に選択（または管理）してから、%iKnow.Source.File.Lister を使用してファイルシステムからドメインヘファイルを直接ロードできます。これはまた、ファイル・リスタを通して以前にロードされたファイルに対応する行を持つ CSV (comma-separated values) ファイルからメタデータ値をロードする機能も提供します。この操作を実行すると、以前は存在しなかったメタデータ・フィールドがその場で自動的に作成されます。

ドメインおよび構成オブジェクトの詳細は、“[NLP 環境の設定](#)” の章を参照してください。リストおよびロードの詳細は、“[プログラムによるテキスト・データのロード](#)” の章を参照してください。

18.5 ドメイン・エクスプローラ

これは、幅広い用途を持つ、サンプル Web ページ・クエリ表示インタフェースです。ここには、エンティティ、CRC、CC、およびパスを含めて、NLP によって識別された各種言語要素に関する豊富な情報が表示され、データに含まれるものをコンテキストの観点から見た概要が表示されます。一般的なフィルタ・オプションを使用すると、メタデータ条件に基づいてドメインのサブセットを容易に選択できます。また、要約オプションを使用すると、ソース自体のコンテンツに速やかにアクセスできます。このインタフェースは、NLP スマート・インデックス作成を使用して大量のドキュメントの概観と移動をすばやく行う方法のサンプルを示しています。

ドメイン・エクスプローラでは、それぞれの頻度や分散と共に上位エンティティ、類似エンティティ、および関連エンティティを含めて、さまざまな NLP クエリ API を呼び出す簡単な例を提供しています。

ドメイン・エクスプローラでは、[skiplist](#) の使用をサポートしています。

機能の詳細は、このドキュメントの“ドメイン・アーキテクト”の章の“[ドメイン・エクスプローラ](#)”を参照してください。

18.6 基本ポータル

このサンプル Web ページ・クエリ表示インタフェースは、ドメイン・エクスプローラの簡素化バージョンです。ここではエンティティとソースのみを表示します。CRC、CC、およびパスは表示しません。これはフィルタ処理および要約機能を提供しますが、既定では、ソースのフル・テキストを表示します。

基本ポータルでは、[skiplist](#) の使用をサポートしています。

18.7 インデックス作成結果

このサンプル Web ページ・クエリ表示インタフェースは 1 つのドキュメントのスマート・インデックス作成結果をチェックして、NLP 分析エンジンが実行した分析の正確さを検証します。これは NLP が各文を一連の概念 (太字とハイライト)、関係 (下線)、および無関係 (イタリック) にどのように分断するかを示します。このページはまた、検出されたコンセプトと CRC を頻度で並べ替えたリストも表示します。このページは入力を手動でロードするためのオプションを提供します。このページは、カスタムのハイライト表示を行うための `%iKnow.Queries.SentenceAPI.GetParts()` の使用方法の例を示しています。

機能の詳細は、このドキュメントの“ドメイン・アーキテクト”の章の“[インデックス作成結果](#)”を参照してください。

概念、関係、CRC、および無関係の詳細は、“[コンセプトの概要](#)”の章を参照してください。

18.8 マッチング結果

このサンプル Web ページ・クエリ表示インタフェースは、各ドキュメントについて、ドキュメントのさまざまなマッチング結果の概要を示します。これにより、ディクショナリとの一致結果に簡単に目を通して、個々の一致の詳細を表示できます。これは `%iKnow.Matching.MatchingAPI.GetHighlightedSentences()` を使用してディクショナリと一致したテキストを色付きのハイライトで表示し、`%iKnow.Matching.MatchingAPI.GetMatchElements()` を使用して特定の一致の詳細 (ディクショナリ名、その項目と用語、一致スコア、一致タイプ、および一致したエンティティ) を表示します。NLP スマート・マッチングを使用して、事前に定義された情報をスマート・インデックス作成結果と結合する方法のサンプルを、このインタフェースは示しています。

マッチングの詳細は、“[スマート・マッチング：ディクショナリの使用](#)”の章を参照してください。

19

InterSystems IRIS 自然言語処理 (NLP) ツール

ドメインを作成してデータを入力するためのプライマリ NLP インタフェースは、[ドメイン・アーキテクト](#)です。NLP データを分析するための主要なインタフェースには、ObjectScript プログラムから呼び出すことが可能なクラス・オブジェクト・メソッドとプロパティを含む API を使用します。この章で説明する NLP ツールは、NLP 機能の管理や NLP データのテストと検証を支援するためのものです。ここで説明するすべての NLP 機能は、iKnow API を使用して利用することもできます。

この章で説明する NLP ツールは、以下のとおりです。

- ・ [シェル・インタフェース](#)
- ・ [データ・アップグレード・ユーティリティ](#)

19.1 NLP シェル・インタフェース

InterSystems IRIS 自然言語処理シェルを使用して、既存のドメインおよびインデックスが作成されているソースに関する情報を返すことができます。

すべての NLP 処理は、ネームスペース内で行われます。したがって、NLP シェルを呼び出す前に、`$namespace` 変数を使用して目的のネームスペースを指定する必要があります。

ターミナルから、次のようにして NLP シェル・インタフェースを有効にできます。

```
USER>DO $System.iKnow.Shell()
```

NLP シェル・プロンプトが返されます。NLP シェル・プロンプトで `?` を入力すると、NLP シェル・コマンドのリストが表示されます。NLP シェル・コマンドの後に空白と `?` を入力すると、そのコマンドに関する情報が表示されます。

NLP シェル・コマンドとオプションは、大文字と小文字が区別されるため、すべて小文字で指定する必要があります。

19.1.1 ソースのリスト、表示、および要約

次の NLP シェルの例では、現在のネームスペースにある “mydomain” という名前の既存のドメインを利用します。ドメイン “mydomain” には 100 個のソースが含まれています。use domain mydomain コマンドは、“mydomain” を NLP シェルの現在のドメインにすることを指定します。list source コマンドは、現在のドメイン内のソースをリストします。既定により、list コマンドは 10 項目のページ・サイズにて、最初のページをリストします。追加のページをリストするには、以下の例で示すように、`>` コマンドを指定することができます。20 項目のページ・サイズに変更するには、use pagesize 20 を指定します。

show source 92 コマンドは、ソース 92 のコンテンツを文に分割してリストします。これは最初のページに 10 個の文をリストします。> コマンドを使用すれば、文の追加ページをリストすることができます。各文のテキストの前後には、文 ID と、表示される文テキストを切り詰める必要があったかどうかを示すブーリアン値が表示されます。(この例では、文のテキストに行の折り返しを追加しています。)show summary 92 6 コマンドは、ソース 92 ~ 6 の文の内容を要約し、これら 6 つの文を表示します。要約番号がソース内の文の数より大きい場合、show summary はソース内のすべての文を表示します。

```
USER>DO $System.iKnow.Shell()
```

```
Welcome to the iKnow shell
Type '?' for help
Type 'quit' to exit
```

```
iKnow> use domain mydomain
Current domain: mydomain (1)
```

```
iKnow> list source
srcId      externalId
100      :SQL:Accident:96
99       :SQL:Accident:98
98       :SQL:Accident:94
97       :SQL:Accident:100
96       :SQL:Accident:80
95       :SQL:Accident:99
94       :SQL:Accident:95
93       :SQL:Accident:88
92       :SQL:Accident:97
91       :SQL:Incident:90
```

```
iKnow> >
srcId      externalId
90       :SQL:Accident:93
89       :SQL:Accident:92
88       :SQL:Accident:91
87       :SQL:Accident:85
86       :SQL:Accident:86
85       :SQL:Accident:89
84       :SQL:Accident:87
83       :SQL:Accident:83
82       :SQL:Accident:78
81       :SQL:Accident:82
```

```
iKnow> show source 92
sentId      sentenceValue
sentenceIsTruncated
5090      On March 7, 2001, about 1500 Alaska standard time, a wheel/ski equipped
Cessna 180 airplane, N9383C, sustained substantial damage during takeoff from
a snow-covered area at Ophir, Alaska. 0
5091      The airplane was being operated as a visual flight rules (VFR) cross-country
personal flight to McGrath, Alaska, when the accident occurred. 0
5092      The airplane was operated by the pilot. 0
5093      The commercial certificated pilot, and the sole passenger, were not injured. 0
5094      Visual meteorological conditions prevailed. 0
5095      During a telephone conversation with the National Transportation Safety Board
(NTSB) investigator-in-charge (IIC), on March 8, 2001, the pilot reported he
landed near Ophir earlier in the day. 0
5096      When he was planning to depart, the surface of the snow had become crusty. 0
5097      The pilot said he began a takeoff run toward the south, but the airplane
did not become airborne until it was within about 50 yards from several trees. 0
5098      During the initial climb, the left horizontal stabilizer collided with a
spruce tree about 25 feet above the ground. 0
5099      The airplane began a descending left turn toward the ground, and collided
with several trees while the pilot was making an emergency landing. 0
```

```
iKnow> show summary 92 6
sentId      sentenceValue
sentenceIsTruncated
5090      On March 7, 2001, about 1500 Alaska standard time, a wheel/ski equipped
Cessna 180 airplane, N9383C, sustained substantial damage during takeoff from
a snow-covered area at Ophir, Alaska. 0
5091      The airplane was being operated as a visual flight rules (VFR) cross-country
personal flight to McGrath, Alaska, when the accident occurred. 0
5095      During a telephone conversation with the National Transportation Safety Board
(NTSB) investigator-in-charge (IIC), on March 8, 2001, the pilot reported he
landed near Ophir earlier in the day. 0
5097      The pilot said he began a takeoff run toward the south, but the airplane
did not become airborne until it was within about 50 yards from several trees. 0
5099      The airplane began a descending left turn toward the ground, and collided
with several trees while the pilot was making an emergency landing. 0
5100      The airplane received damage to the left main landing gear, the wings,
and the left stabilizer. 0
```

```
iKnow> quit
```


Bye bye

USER>

19.1.2 ソースのフィルタ処理

次の NLP シェルの例では、現在のネームスペースにある “mydomain” という名前の既存のドメインを利用します。ドメイン “mydomain” には 100 個のソースが含まれています。use domain mydomain コマンドは、“mydomain” を NLP シェルの現在のドメインにすることを指定します。list source コマンドは、現在のドメインで最初の 10 個のソースをリストします。filter source 92 94 97 as myfilter コマンドは、ソース ID で指定されたソース以外のすべてのソースを除外する “myfilter” という名前のフィルタを定義します。use filter myfilter コマンドは、“myfilter” を現在のフィルタに設定します。ここで、NLP シェルから list source コマンドを発行すると、“myfilter” が適用され、“myfilter” で指定された 3 つのソースだけがリストされます。

```
USER>DO $System.iKnow.Shell()

Welcome to the iKnow shell
Type '?' for help
Type 'quit' to exit

iKnow> use domain mydomain
Current domain: mydomain (1)
iKnow> list source
srcId      externalId
100       :SQL:Accident:96
99        :SQL:Accident:98
98        :SQL:Accident:94
97        :SQL:Accident:100
96        :SQL:Accident:80
95        :SQL:Accident:99
94        :SQL:Accident:95
93        :SQL:Accident:88
92        :SQL:Accident:97
91        :SQL:Incident:90
iKnow> filter source 92 94 97 as myfilter
iKnow> use filter myfilter
Current filter: myfilter

iKnow> list source
srcId      externalId
97         :SQL:Accident:100
94         :SQL:Accident:95
92         :SQL:Accident:97
iKnow> quit
Bye bye

USER>
```

フィルタが適用されると、後続の use filter filtername によって、現在のフィルタが新しいフィルタに置き換えられます。フィルタを無効にするには、use filter 0 を指定します。

19.2 NLP データ・アップグレード・ユーティリティ

NLP データ構造の各バージョンには、システム・バージョン番号が割り当てられています。各 NLP ドメインには、Version プロパティの値が割り当てられます。すべての新規ドメインは、現在のシステム・バージョンとして同じ Version プロパティを使用して作成されます。したがって、通常これら 2 つの整数値は同じになります。

- ・ InterSystems IRIS® Data Platform を新しいバージョンに更新したときに、NLP データ構造のシステム・バージョンがインクリメントされるのは、NLP インデックス作成が変更された場合です。したがって、InterSystems IRIS を新しいバージョンに更新しても、必ずしも NLP データ構造のシステム・バージョンがインクリメントされるとは限りません。
- ・ ドメインを作成すると、そのドメインは現在のデータ構造のシステム・バージョンを Version プロパティの値として取得します。したがって、以前のシステム・バージョンで作成された既存のドメインは、その以前のシステム・バージョンを Version プロパティの値として持っています。

最初の InterSystems IRIS のバージョン番号は 5 です。既存のドメインを InterSystems IRIS に移植すると、これより前のバージョン番号が表示される場合があります。

`%iKnow.Domain` クラスの `GetCurrentSystemVersion()` メソッドを使用して、現在の InterSystems IRIS インスタンスの NLP データ構造のシステム・バージョンを判定できます。“NLP 環境の設定”の章の“[全ドメインのリスト](#)”で説明したとおり、`GetAllDomains` クエリを使用して、すべてのドメインのドメイン・バージョン番号をリストできます。

NLP データ構造のシステム・バージョンがドメイン・バージョンと一致しない場合、それらの古い NLP ドメインでは、この新しいシステム・バージョンで導入された NLP の新機能やパフォーマンスの改善を利用できません。古いドメインは引き続き操作可能ですが、ドメインをアップグレードするまでは NLP データ構造の新機能を利用できません。ドメインをアップグレードすると、ドメインの `Version` プロパティがインクリメントされます。このアップグレード処理では、ドメイン・データのインデックスを自動的に再作成する必要があります。元のソース・テキストにアクセスする必要はありません。各ドメインは個々にアップグレードする必要があります。

ドメインをアップグレードするには、`%iKnow.Utils.UpgradeUtils` クラスの `UpgradeDomain()` メソッドを使用します。詳細は、“[インターシステムズ・クラス・リファレンス](#)”ドキュメントを参照してください。

ドメインのアップグレード時に行われるインデックス再作成により、ドメイン ID は変更されますが、ドメイン名は変更されません。したがって、ドメインをアップグレードすると、特定のドメイン ID の整数値によってドメインを参照するプログラムに対しては変更が必要な場合があります。このため、ドメインを常にその ID プロパティ (またはドメイン名) で参照することをお勧めします。リテラルの整数 ID 値でドメインを参照するコーディング方法は避けてください。

20

iKnow Web サービス

%iKnow パッケージは、NLP クエリを実行する Web サービス・クラスを提供します。この章では、これらのクラスの概要と使用法を説明します。以下のトピックについて説明します。

- ・ [利用可能な NLP Web サービス](#)
- ・ [NLP Web サービスの使用法](#)
- ・ [例](#)
- ・ [NLP Web サービスと主要 API の比較](#)
- ・ [情報のその他の参照先](#)

20.1 利用可能な Web サービス

NLP は以下の Web サービス・クラスを提供します。

- ・ **%iKnow.Queries.CcWSAPI** – コンセプト – コンセプト (CC) ペアに関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.CcAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.CrcWSAPI** – コンセプト – リレーション – コンセプト (CRC) という 3 個組に関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.CrcAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.EntityWSAPI** – エンティティに関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.EntityAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.EquivWSAPI** – 同義性の管理に使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.EquivAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.MetadataWSAPI** – ソース・メタデータに関する情報を管理および取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.MetadataAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.PathWSAPI** – パスに関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.PathAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.SentenceWSAPI** – 文に関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.SentenceAPI** クラスのメソッドに相当します。
- ・ **%iKnow.Queries.SourceWSAPI** – ソースに関する情報を取得するために使用できる Web メソッドを提供します。これらの Web メソッドは、**%iKnow.Queries.SourceAPI** クラスのメソッドに相当します。

%iKnow.Queries パッケージには、これらのクラスがコンパイルされたときにコンパイラによって生成されたクラスも含まれます。例えば、コンパイラはクラス %iKnow.Queries.CcWSAPI をコンパイルしたときに、%iKnow.Queries.CcWSAPI パッケージにクラスを生成しています。生成されたこれらのクラスは、直接使用するためのものではありません。

20.2 NLP Web サービスの使用法

Web サービスを使用するには、それと通信する Web クライアントを作成して使用します。NLP Web サービスの場合も、他の Web サービスの場合と同じ手順でこれを行います。

1. Web サービスと通信できる Web クライアントを作成します。通常、これを行うには、クライアント・テクノロジーが提供するツールを使用して Web クライアントを生成し、Web サービスの WSDL を入力として提供します。このプロセスによって、一連のクライアント・クラスが生成されます。

InterSystems IRIS® Data Platform Web サービスの場合、以下の URL で WSDL を取得できます。

```
http://hostname:port/csp/namespace/web_service_class.cls?WSDL
```

以下はその説明です。

- ・ hostname は、InterSystems IRIS が実行されているサーバです。
- ・ port は、Web サーバが実行されているポートです。
- ・ namespace はネームスペース名です。
- ・ web_service_class は、.cls で終わる Web サービスの完全なパッケージおよびクラス名です。

例えば、クラス %iKnow.Queries.EntityWSAPI の場合は、%25iKnow.Queries.EntityWSAPI.cls を使用します。

重要 ここに示すように、パッケージの先頭のパーセント記号を、必ず URL エスケープ・シーケンスの %25 で置き換えてください。

以下に例を示します。

```
http://localhost:52773/csp/samples/%25iKnow.Queries.EntityWSAPI.cls?WSDL
```

2. 生成されたクライアント・クラスを編集するのではなく、それらを使用する追加クラスやルーチンを作成します。詳細は使用するテクノロジーによって異なります。

InterSystems IRIS では、Web クライアントを使用するには、Web クライアント・クラスのインスタンスを作成し、次にそのインスタンス・メソッドを呼び出します。以下の例を参照してください。

20.3 例

デモンストレーションのため、同じマシン上で NLP Web サービスと共に動作する InterSystems IRIS Web クライアントを生成して使用します。SAMPLES ネームスペースでこれを行うには、以下の手順に従います。

1. スタジオで、[ツール]→[アドイン]→[SOAP ウィザード] をクリックします。
2. 最初の画面で、[URL] をクリックします。

- 以下の URL を入力します。

```
http://localhost:52773/csp/samples/%25iKnow.Queries.EntityWSAPI.CLS?WSDL
```

必要な場合は、Web サーバがこの InterSystems IRIS インストールに使用しているポートで 52773 を置き換えます。

- [次へ] をクリックします。
- [プロキシクラスパッケージ] に MyClient のようなパッケージ名を入力します。
- [次へ] を 2 回クリックします。

ウィザードによって、クラスが生成およびコンパイルされ、これらのクラスの一覧が表示されます。

このプロセスによって、クラス MyClient.iKnow.Queries.EntityWSAPISoap が生成されます。このクラスは、このクライアントが基づく Web サービスで定義された各 Web メソッドに 1 つずつ、一連のメソッドを定義します。これらの各メソッドは以下のようになります (ここでは読みやすいように改行を追加してあります)。

```
Method GetByFilter(domainid As %Integer, filter As %String, filtermode As %Integer, enttype As
%Integer,
skipListIds As
%ListOfDataTypes(ELEMENTTYPE="%String",XMLITEMNAME="skipListIdsItem",XMLNAME="skipListIds"))
As %XML.DataSet [ Final, ProcedureBlock = 1, SoapBindingStyle = document, SoapBodyUse = literal,
WebMethod ]
{
    Quit ..WebMethod("GetByFilter").
}

Invoke($this,"http://www.intersystems.com/iKnow/Queries/EntityWSAPI/%iKnow.Queries.EntityWSAPI.GetByFilter",
        .domainid,.filter,.filtermode,.enttype,.skipListIds)
}
```

このメソッドのシグニチャは、Web サービスの対応するメソッドのシグニチャと同じになります。

このプロセスはまた、MyClient.iKnow.Queries.EntityWSAPISoap に、直接使用することはできない一連のクラスを生成します。

- [完了] をクリックします。
- 生成されたクライアントを使用するには、ターミナルで SAMPLES ネームスペースに以下のように入力します。

```
set client=##class(MyClient.iKnow.Queries.EntityWSAPISoap).%New()
```

これによって Web サービスと通信できるクライアント・クラスのインスタンスが作成されます。

- このクラスのメソッドを実行します。以下はその例です。

```
write client.GetCountByDomain(2)
```

20.4 NLP Web サービスと主要な NLP API の比較

SOAP メッセージで簡単に表現できない引数を主要な NLP API は使用するため、主要な NLP API のメソッドとは異なるシグニチャを NLP Web サービスのメソッドは持ちます。特に、次の相違点に注意してください。

- %Library.List の代わりに、Web サービス (およびそのクライアント) は %Library.ListOfDataTypes を使用します。つまり、Web サービスのリストを作成する際は、%Library.ListOfDataTypes のインスタンスを作成してから、その SetAt() メソッドを使用して項目を追加します。このインスタンスでは \$LISTBUILD のようなリスト関数は使用できません。
- %iKnow.Filters.Filter のインスタンスの代わりに、Web サービス (およびそのクライアント) は、そのクラスの ToString() メソッドから返された形式の文字列を使用します。

- ・ 複雑な結果を返す API の場合は、多次元配列として参照により結果を返す代わりに、Web サービス (およびそのクライアント) は `%XML.DataSet` のインスタンスを返します。

重要 この構造は .NET およびインターシステムズの製品でのみサポートされています。他の Web テクノロジは、この形式を認識しません。

20.5 関連項目

InterSystems IRIS での Web サービスおよびクライアントの詳細は、“[Web サービスおよび Web クライアントの作成](#)”を参照してください。

21

KPI とダッシュボード

InterSystems IRIS® Data Platform ダッシュボード・テクノロジーでは、エンド・ユーザ向けの Web ベース・ダッシュボードを作成できます。他の項目としては、ダッシュボードに KPI (重要業績評価指標) を表示できます。通常、KPI は、ダッシュボード上で実行および表示が可能なクエリで、ダッシュボードが表示されたときに、クエリが実行されます。

このセクションでは、テキスト分析に基づいて KPI を作成する方法とダッシュボードに KPI を表示する方法について説明します。

キューブ内でのテキスト分析データ使用の詳細は、“InterSystems Business Intelligence の上級モデリング” を参照してください。

21.1 KPI 用語

インターシステムズの KPI では、クエリによって返される各行は、KPI の独立した系列になります。以下の図は、KPI のいくつかの系列を示しています (この章で後述する KPI テスト・ページに表示されています)。系列名は、[KPI 値] テーブルの最初の列に示されます。

KPI

Class	GIKNOW.TopEntitiesKPI
Name	TopEntities
Caption	TopEntities

Filters 5 filter(s)

IK:PARAM:NAMECOLUMN	IK:QPARAM:4:PAGE SIZE	IK:QPARAM
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Submit Query"/>		

KPI Values 10 series

Series	resultNumber	entity	entUnild	frequency	spread
1	1	researchers	2915	31	15
2	2	animals	340	25	12
3	3	study	3287	20	10
4	4	birds	574	16	6
5	5	news	2393	16	9

(既定では) 系列名は、クエリが返した最初の列の値になります。

KPI には、プロパティも含まれ、そのそれぞれが返されるデータの列に対応します。前述の例では、KPI は 5 つのプロパティを持っています。KPI を定義する際に、これらのプロパティ名とその順序をオーバーライドできます。

KPI クエリにはパラメータを含めることができます。これらは KPI フィルタと呼ばれます。なぜなら、これらは通常 (ただし、常時ではありません)、KPI が返した値をフィルタするからです。KPI テスト・ページには、利用可能な KPI フィルタがすべて表示されます。KPI メカニズムは、任意の KPI に対して一連の KPI フィルタを自動的に提供します。

また、KPI は、カスタム・コードを実行するアクションも定義できます。KPI をダッシュボードに追加する際は、これらのアクションを実行するためのコントロールをダッシュボードに追加できます。ページングを使用するテキスト分析クエリに基づく KPI の場合、その KPI では [前のページ] および [次のページ] アクションが定義されます。

21.2 テキスト分析クエリを使用する KPI の定義

テキスト分析クエリを使用する KPI を定義するには、以下の手順を実行します。

- 以下のクラスの 1 つを拡張するクラスを作成します。
 - `%iKnow.DeepSee.GenericKPI` – ほとんどのクエリには、これを使用します。次の手順でクエリを指定します。
 - `%iKnow.DeepSee.SourceListFilterKPI` – これを使用して、ソースに関する情報を表示します。この場合は、システムは `%iKnow.Queries.SourceAPI` の `GetByEntities()` または `GetByDomain()` メソッドを使用します。
- このクラスでは、使用するテキスト分析ドメインを指定します。これを実行するには、以下のいずれかを実行します。
 - ドメインの整数 ID を指定します。これを実行するには、`IKDOMAINID` クラス・パラメータをオーバーライドして、テキスト分析ドメインの整数 ID に設定します。
 - ドメインを定義するキューブと NLP (自然言語処理) メジャーを指定します。これを実行するには、以下のクラス・パラメータをオーバーライドします。
 - `IKCUBENAME` : キューブの論理名と同じです。
 - `IKMEASURENAME` : 指定されたキューブ内の NLP メジャーのローカル名と同じです。

この手法は、キューブの統合を使用している場合にのみ可能です。詳細は、“InterSystems Business Intelligence の上級モデリング” を参照してください。
- また、使用したスーパークラスに応じて、以下のクラス・パラメータをオーバーライドします。
 - `IKPAGESIZE` – 任意のページに表示される行の数と等しくなります。既定値は 10 です。
ダッシュボード内で、これはダッシュボード・ウィジェットでページごとに表示される行数に影響を与えます。
 - `IKQUERYCLASS` – (`%iKnow.DeepSee.GenericKPI` のサブクラスを作成する場合のみ) API の完全なパッケージおよびクラス名と等しくなります。ObjectScript API に属する API クラスの 1 つを使用します。例：
`%iKnow.Queries.EntityAPI`
 - `IKQUERYNAME` – (`%iKnow.DeepSee.GenericKPI` のサブクラスを作成する場合のみ) そのクラスのメソッド名と等しくなります。
- 以下のような XData ブロックをクラスに追加します。これにより、KPI の論理名と表示名を指定します。

```
/// This XData definition defines the KPI.
XData KPI [ XMLNamespace = "http://www.intersystems.com/deepsee/kpi" ]
{
  <kpi name="MyKPI" displayName="My KPI">
  </kpi>
}
```

必要に応じて `name` および `displayName` の値を指定します。`name` にはスペースや句読点を含めることはできません。`displayName` の値はローカライズ可能です。`displayName` は省略することができ、既定では `name` が使用されます。

後述のサブセクション “[KPI プロパティのオーバーライド](#)” も参照してください。

5. クラスをコンパイルします。

オプションとして KPI テスト・ページを使用します。(現在スタジオでクラスを表示している場合は、[表示]→[ウェブページ] をクリックします。)

KPI をテストするには、[フィルタ] セクションのドロップダウン・リストを使用します。次に [クエリ送信] をクリックします。

これらのドロップダウン・リストには、利用可能な KPI フィルタがすべて表示されます。次のサブセクションを参照してください。

21.3 利用可能な KPI フィルタ

KPI のクエリには、パラメータを含めることができます。こうしたパラメータは KPI フィルタと呼ばれます。なぜなら、これらは通常 (ただし、常時ではありません)、KPI が返した値をフィルタするからです。テキスト分析に基づく KPI では、以下の KPI フィルタが自動的に提供されます。

- ・ 使用しているクエリに対して意味のあるクエリ・パラメータ・サブセット (テキスト分析ドメイン ID といった一部のクエリ・パラメータは自動的に処理され、公開されません)
- ・ このドメインのすべてのパブリック・ソース・メタデータ・フィールド
- ・ 系列名として使用する列を指定するための NAMECOLUMN パラメータ

重要

フィルタ適用後に実行する統計的な再処理の指定に使用する `filtermode` 引数がクエリの多くに含まれています。詳細は、このドキュメントに前述されている “[フィルタ・モード](#)” を参照してください。このようなクエリを直接使用する場合、既定の `filtermode` は、処理をまったく実行しない `$$$FILTERONLY` になります。

こうしたクエリを KPI として公開すると、`filtermode` 引数は常に `$$$FILTERALLANDSORT` と指定されるため、頻度と分散の統計は再計算されて、結果は再度並べ替えられます。したがって、`filtermode` の指定方法により、こうしたクエリを KPI として使用する場合は、これらを直接使用した場合とは結果が異なる可能性があります。

21.4 KPI プロパティのオーバーライド

既定では、`%Know.DeepSee.GenericKPI` に基づく任意の KPI は、クエリ結果と同じ名前を使用し、クエリ結果と同じ順序ですべての結果列を公開します。この順序や名前は変更することができ、列は非表示にすることができます。

それぞれの結果列は KPI プロパティです。ユーザがダッシュボードを作成すると、以下が表示されます。

- ・ 既定では、ユーザが KPI に基づくピボット・テーブル・ウィジェットを追加すると、KPI で定義されているものと同じ順序と名前プロパティが表示されます。
- ・ KPI に基づくスコアカード・ウィジェットを追加する場合は、ユーザが表示するプロパティを選択します。選択肢のドロップダウン・リストには、KPI で定義されているものと同じ順序と名前のプロパティが含まれます。
- ・ いずれの場合も、ユーザは KPI プロパティの順序や KPI プロパティに対して表示されるタイトルを変更できます。

KPI プロパティの順序や名前を変更したり、プロパティを非表示にするには、次のように XData ブロック内に一連の `<property>` 要素を追加します。

Class Member

```
XData KPI [ XMLNamespace = "http://www.intersystems.com/deepsee/kpi" ]
{
<kpi name="MyKPI" displayName="My KPI">
  <property name="entity" displayName="Entity" />
  <property name="frequency" displayName="Frequency" />
  <property name="spread" displayName="Spread" />
</kpi>
}
```

`<property>` では、`name` の値は、クエリが返すフィールドの名前と正確に一致する必要があります。`displayName` には、ユーザに表示される名前を使用します。`<property>` 要素を希望の順序でリストし、表示するプロパティをすべてリストします。

21.5 例

以下に KPI の例を示します。

Class Definition

```
Class GIKNOW.TopEntitiesKPI Extends %iKnow.DeepSee.GenericKPI
{
Parameter IKDOMAINID = 1;
Parameter IKPAGE SIZE As %Integer = 10;
Parameter IKQUERYCLASS = "%iKnow.Queries.EntityAPI";
Parameter IKQUERYNAME = "GetTop";

/// This XData definition defines the KPI.
XData KPI [ XMLNamespace = "http://www.intersystems.com/deepsee/kpi" ]
{
<kpi name="TopEntities" displayName="Top Entities in text analytics domain 1" >
  <property name="resultNumber" displayName="rank"/>
  <property name="entity" displayName="entity"/>
  <property name="entUniId" displayName="Id"/>
  <property name="frequency" displayName="frequency"/>
  <property name="spread" displayName="spread"/>
</kpi>
}
}
```

別の例は、この次のセクションの“[KPI でのダッシュボードの例](#)”を参照してください。

21.6 KPI を表示するダッシュボードの作成

KPI をユーザに利用可能にするには、KPI をダッシュボードに追加します。ユーザは、さまざまな方法でダッシュボードにアクセスできるようになります。詳細は、この章で後述される“[ダッシュボードへのアクセス](#)”を参照してください。

- ・ [ダッシュボードを作成する方法](#)
- ・ [系列名の変更](#)
- ・ [プロパティの構成](#)

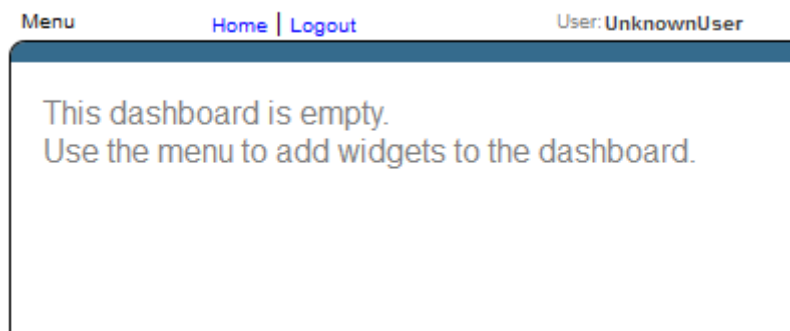
- ・ [\[前のページ\]](#) および [\[次のページ\]](#) ボタンの追加
- ・ [最大表示長の設定](#)
- ・ [ダッシュボードの例](#)


21.6.1 ダッシュボードの作成 : Basics

以下の手順は、KPI を表示するシンプルなダッシュボードの作成方法を簡単に説明しています。

1. 管理ポータルで、[Analytics] > [ユーザ・ポータル] をクリックしてから、[ビュー] をクリックします。
ユーザ・ポータルが表示され、既存のパブリック・ダッシュボードとピボット・テーブルがこのネームスペースにリストされます。
2. [メニュー]→[新しいダッシュボード] をクリックします。
新しいダッシュボードに関する基本情報の入力を求めるダイアログ・ボックスが表示されます。
3. [ダッシュボード名] に値を入力します。
4. オプションで [フォルダ] の値を指定します。
5. [ページレイアウト] では、3 番目のオプション (ワークボックスなし) をクリックします。
6. [OK] をクリックします。

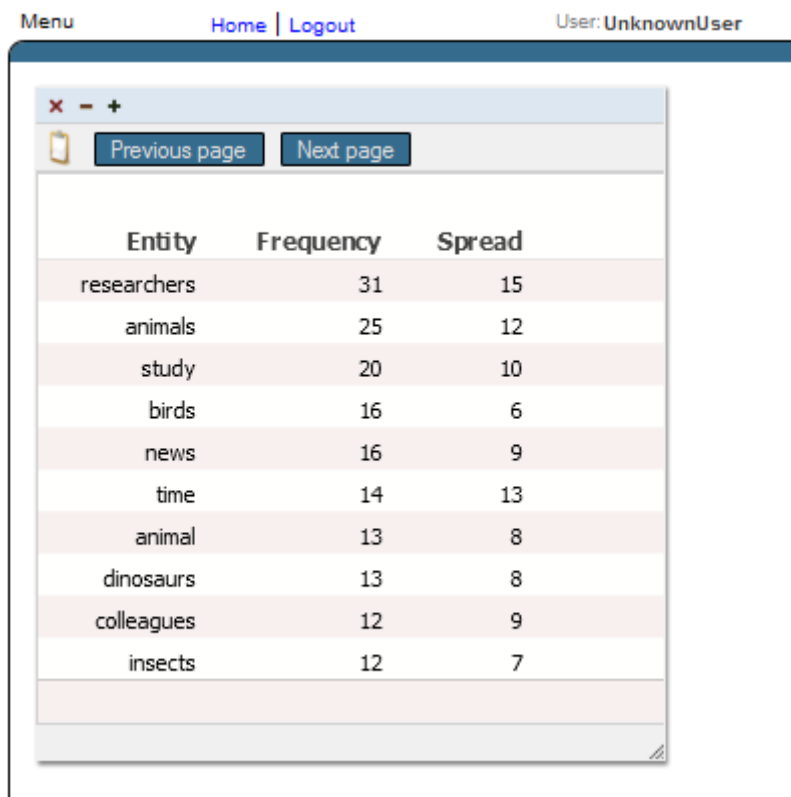
ダッシュボードの作成、保存、および表示が行われます。ダッシュボードは最初は空です。



7. ウィジェットを追加して KPI を表示します。そのためには、以下の操作を実行します。
 - a. [メニュー]→[新規ウィジェット追加...] をクリックします。
 - b. 左の領域で、項目をクリックして選択肢リストを表示します。
ピボット・テーブル・ウィジェットまたはスコアカードのいずれかを選択します。
 - c. 使用するウィジェットのタイプをクリックします。
 - d. [データソース] タブで、[データソース] の横にある検索ボタン  をクリックします。
 - e. KPI の名前をクリックします。
 - f. [OK] をクリックします。
 - g. スコアカードを選択した場合は、“[プロパティの構成](#)” を参照してください。
 - h. [OK] をクリックして、ウィジェットを追加します。
8. ウィジェットのサイズを変更し、もう一度 [メニュー]→[保存] をクリックします。
9. 構成オプションの詳細は、以下のサブセクションを参照してください。

結果は以下のようになります。

Menu Home | Logout User: UnknownUser

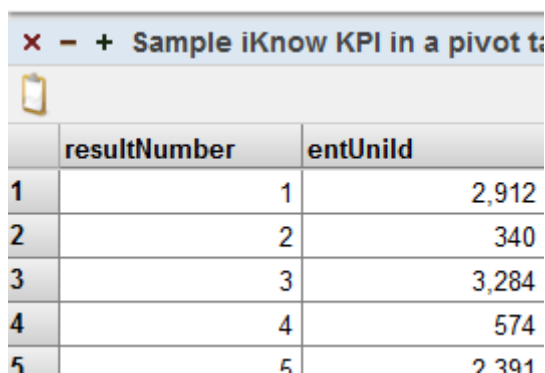


Entity	Frequency	Spread
researchers	31	15
animals	25	12
study	20	10
birds	16	6
news	16	9
time	14	13
animal	13	8
dinosaurs	13	8
colleagues	12	9
insects	12	7

この章の[最後のセクション](#)に、追加情報へのリンクが示されています。



21.6.2 系列名の変更


KPI をピボット・テーブル・ウィジェットに表示すると、行の名前として系列名が使用されます。既定では、戻り値の最初の列が系列名に使用されますが、これはこのシナリオでは役に立たないかもしれません。例えば、以下の KPI では、最初の列は結果番号です。



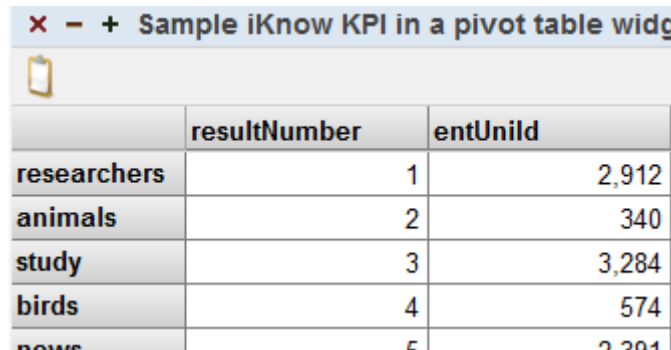
	resultNumber	entUnild
1	1	2,912
2	2	340
3	3	3,284
4	4	574
5	5	2,391

系列名として別の列を選択するには、以下の手順に従います。

1. ウィジェットの再構成ボタン  をクリックします。
2. [コントロール] タブをクリックします。
3. テーブルの右側の追加ボタン  をクリックします。このテーブルは最初は空です。
4. [タイプ] で [隠し] を選択します。

5. [アクション] で、[フィルタ] を選択します。
6. [フィルタ] で、[系列名列] を選択します。
7. [デフォルト値] で、追加ボタン  をクリックします。
8. 使用する列の名前を選択します（例えば、ここで示された KPI の [entity]）。
9. [OK] をクリックして、既定値を追加します。
10. [OK] をクリックして、コントロールを追加します。
11. [OK] をクリックして、ウィジェットの再構成を完了します。

これで行名が変更されました。以下はその例です。





	resultNumber	entUnild
researchers	1	2,912
animals	2	340
study	3	3,284
birds	4	574
books	5	2,201

21.6.3 プロパティの構成

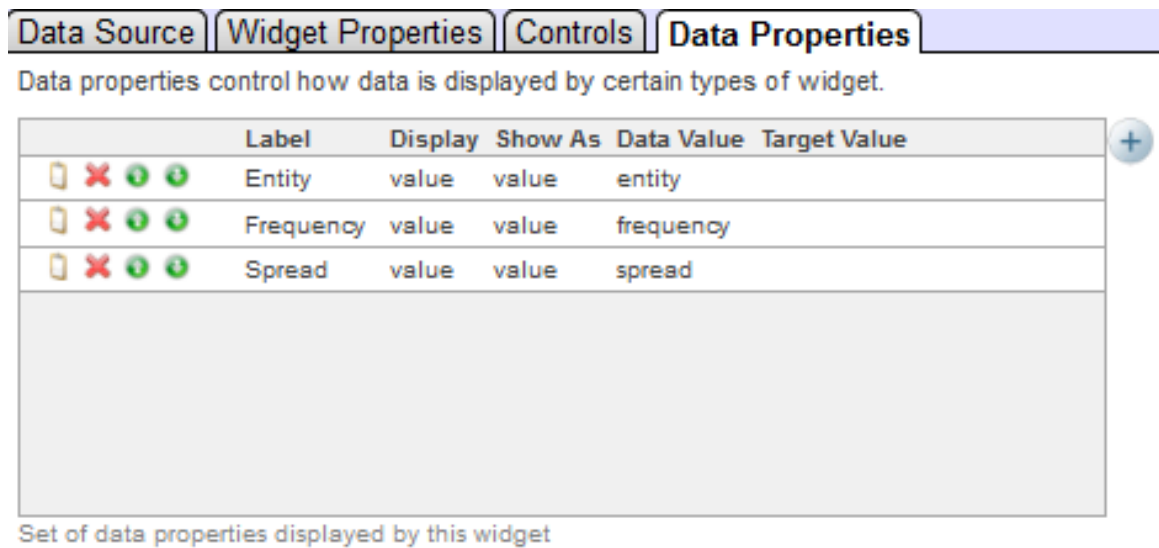
KPI をウィジェットに表示する際は、そのウィジェットに表示されるプロパティの再構成が必要な場合があります。

- ・ スコアカード・ウィジェットの場合は、プロパティを構成する必要があります。既定では、列は表示されません。
- ・ ピボット・テーブル・ウィジェットの場合は、プロパティの構成が必要になる可能性があります。既定では、すべての列が表示されます。

プロパティを構成する手順は次のとおりです。

1. ウィジェットの再構成ボタン  をクリックします。
2. [データのプロパティ] タブをクリックします。
このタブで、KPI のプロパティ（または列）を指定します。
3. プロパティを構成するには、追加ボタン  をクリックします。
4. [データ値] で、ドロップダウン・リストから値を選択します。これには、KPI の結果列がリストされています。
5. オプションとして、[ラベル] にキャプションを入力します。
6. オプションとして、[形式] に形式文字列を入力します。既定では、数値は、ロケールで使用する 1000 単位の区切り文字を使用して表示されます。1000 単位の区切り文字を使用しないで数字の形式にするには、[形式] に「#」を入力します。
7. [OK] をクリックします。

KPI のそれぞれの結果列を追加すると、KPI に応じて、ダイアログ・ボックスは次のようになります。



8. [OK] をクリックして、再構成を完了します。

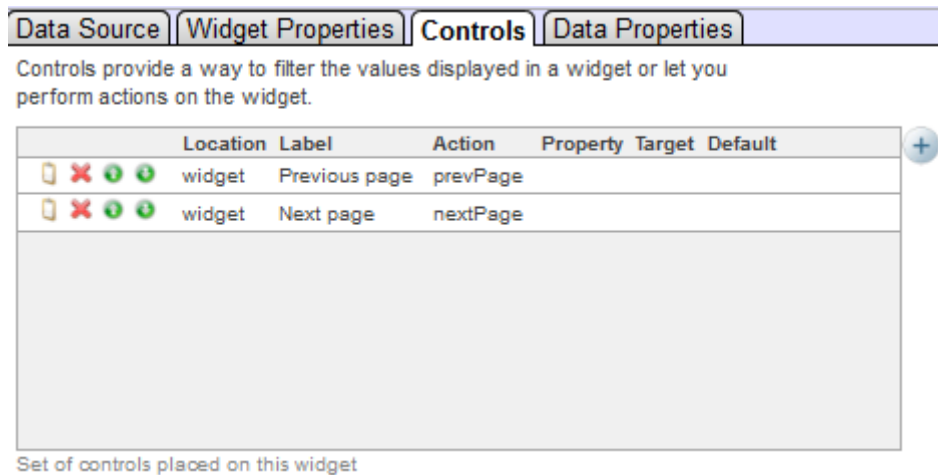
21.6.4 [前のページ] および [次のページ] ボタンの追加

ページングを使用するクエリに基づく KPI の場合、その KPI では [前のページ] および [次のページ] アクションが定義されます。ウィジェットは、これらのアクションを実行するボタンを含むように構成できます。そのためには、以下の操作を実行します。

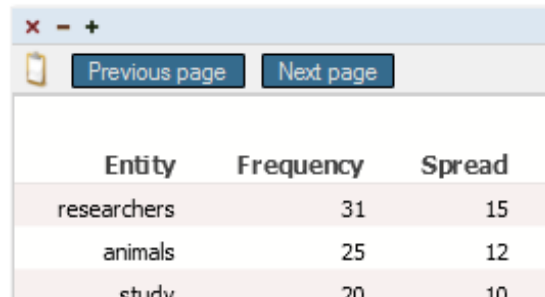
1. ウィジェットの再構成ボタン をクリックします。
2. [コントロール] タブをクリックします。
3. 次のように、[前のページ] ボタンをこのウィジェットに追加します。
 - a. テーブルの右側の追加ボタン をクリックします。このテーブルは最初は空です。コントロールを指定するダイアログ・ボックスが表示されます。
 - b. [アクション] で、[前のページ] を選択します。
 - c. [コントロールのラベルまたはアイコン] に、「」と入力します。
 - d. [OK] をクリックして、コントロールを追加します。

同様の手順を使用して [次のページ] ボタンを追加します。

すると、ダイアログ・ボックスは以下のようになります。



4. [OK] をクリックして、ウィジェットの再構成を完了します。
すると、ウィジェットには次のようなボタンが含まれます。



21.6.5 最大表示長の設定

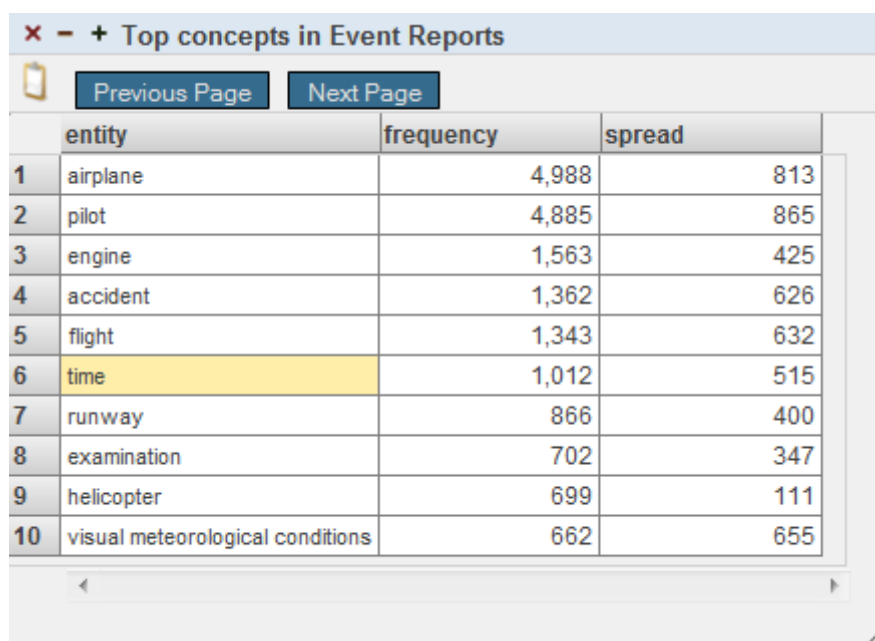
KPI クラスを定義する際、ダッシュボードに表示する最大行数を設定できます。これは、一般的な KPI クラスの拡張で `MAXLISTINGROWS` パラメータを編集して行うことができます。この値は、整数である必要があります。ユーザ・クラスでこのパラメータが未定義の場合は、既定値の 1000 に設定されます。

21.6.6 KPI でのダッシュボードの例

SAMPLES ネームスペースでは、テキスト分析を使用する KPI を表示するダッシュボードの例が用意されています。このダッシュボードを表示するには、以下の手順に進みます。

1. 管理ポータルで、**SAMPLES** ネームスペースにアクセスします。
2. [Analytics] > [ユーザ・ポータル] をクリックしてから、[ビュー] をクリックします。
ユーザ・ポータルが表示され、既存のパブリック・ダッシュボードとピボット・テーブルがこのネームスペースにリストされます。
3. [Aviation event reports] をクリックします。
要求されたダッシュボードがシステムで表示されます。

このダッシュボードで、[イベント・レポートの上位概念] ウィジェットを探します。このウィジェットは以下のように表示されます。



	entity	frequency	spread
1	airplane	4,988	813
2	pilot	4,885	865
3	engine	1,563	425
4	accident	1,362	626
5	flight	1,343	632
6	time	1,012	515
7	runway	866	400
8	examination	702	347
9	helicopter	699	111
10	visual meteorological conditions	662	655

このウィジェットには、`Aviation.KPI.TopConcepts` クラスに定義された KPI が表示されます。

21.7 ダッシュボードへのアクセス

ダッシュボードは、さまざまな方法でユーザに提供できます。

- ・ アプリケーションによってダッシュボードへの直接リンクを提供できます。
- ・ ダッシュボードを表示可能なユーザ・ポータルを作成できます。InterSystems ユーザ・ポータルを使用するか、または独自のポータルを作成できます。

InterSystems ユーザ・ポータルはエンド・ユーザが直接使用することを目的としています。

21.8 関連項目

- ・ ダッシュボードの作成に関する詳細は、“ダッシュボードの作成”を参照してください。
- ・ アプリケーションからダッシュボードへのアクセスに関する詳細は、“InterSystems Business Intelligence の実装”の“アプリケーションからダッシュボードへのアクセス”を参照してください。
- ・ クラスへのダッシュボード定義のパッケージ化に関する詳細は、“InterSystems Business Intelligence の実装”の“クラスへの Business Intelligence 要素のパッケージ化”を参照してください。
- ・ InterSystems ユーザ・ポータルおよびダッシュボードの詳細は、“ダッシュボードとユーザ・ポータルの使用法”を参照してください。

22

NLP のカスタマイズ

NLP 意味分析エンジンが InterSystems IRIS® Data Platform でソース・テキストをロードおよびリストする方法をカスタマイズすることができます。以下のタイプのカスタマイズがサポートされています。

- ・ **リスタ**。既定では、NLP は一般的なデータ・ソースに対応するいくつかのリスタを提供します。一方で、使用するデータに適したカスタム・リスタを作成することも可能です。
- ・ **プロセッサ**。既定では、NLP はリスタに対応するプロセッサを使用します。一方、カスタム・プロセッサを作成してカスタム・リスタと関連付けたり、既存のプロセッサをカスタム・リスタに関連付けることができます。
- ・ **コンバータ**。コンバータは、複雑な入力テキストを NLP エンジン向けのプレーン・テキストに変換することが可能なオブジェクトです。例えば、コンバータは PDF または RTF ドキュメントからプレーン・テキストを抽出したり、XML ドキュメントから特定のノードを選択することができます。既定では、NLP はコンバータを使用しません。ユーザはカスタム・コンバータを作成し、リスタの `SetConverter()` または `Init()` メソッドでこれを指定できます。

リスタでは、オプションとして、その構成、プロセッサ、およびコンバータを指定できます。これらは、`SetConfig()`、`SetProcessor()`、および `SetConverter()` メソッドを使用して指定できます。または、`Init()` メソッドを使用すると、3 つをすべて指定できます。

以下の例は、オプション・リスタの `Init()` メソッドを使用してカスタム・プロセッサやカスタム・コンバータを指定する方法を示しています。どの `Init()` メソッド・パラメータに対しても、空の文字列("") を指定することで、指定のリスタの既定値を使用できます。

```
SET flister=##class(%iKnow.Source.File.Lister).%New(domId)
DO flister.Init(configuration,myprocessor,processorparams,myconverter,converterparams)
```

22.1 カスタム・リスタ

NLP は、ベース・リスタ・クラスに加え、各種入力ソースに固有のリスタを含む 5 つのサブクラスを提供しています。

カスタム・リスタを実装するには、まずベース・リスタ・クラスの `%iKnow.Source.Lister` を使用して、その既定値のいくつかをオーバーライドします。

22.1.1 リスタ名

リスタを使用するには、リスタは各ソースの外部 ID を表す形式を指定する必要があります。リスタの外部 ID は、リスタ名と完全参照から構成されます。完全参照は groupName と localRef から構成されます。以下の例で 外部 ID を示します。

```
:MYLISTER:groupname:localref
```

この例では、MYLISTER がリスタ名のエイリアスになります。エイリアスを指定しないと、リスタ・クラスの完全なクラス名が使用されます。リスタのエイリアスを判別するには、GetAlias() メソッドを使用します。

リスタ名エイリアスは、現在のネームスペース内で重複しないようにします。すでに存在しているリスタ名エイリアスを指定すると、`$$$IKLsterAliasInUse` エラーが生成されます。

22.1.2 SplitFullRef() と BuildFullRef()

カスタム・リスタには SplitFullRef() インスタンス・メソッドを指定する必要があります。このメソッドは、groupName と localRef を fullRef 文字列から抽出するために使用されます。その結果は SplitExtId クラス・メソッドに提供されます。リスタが `:MYLISTER:groupname:localref` のような外部 ID 形式を持つと仮定してみましょう。

このシンプルな例では、fullRef は文字列 groupname:localref から構成されるため、groupName は `$PIECE(fullRef, ":", 1)`、localRef は `$PIECE(fullRef, ":", 2)` になります。これは非常にシンプルな例で、groupName か localRef の部分に ":" 文字が含まれていると機能しないことに注意してください。

カスタム・リスタには BuildFullRef() インスタンス・メソッドを指定する必要があります。このメソッドは、groupName と localRef を組み合わせて fullRef 文字列を構成するために使用されます。その結果は BuildExtId クラス・メソッドに提供されます。

22.1.3 既定のプロセッサ

プロセッサとは、リスタが生成したリストを入力として使用し、対応するソース・テキストを読み取り、インデックス作成のためにそのソース・データを NLP エンジンに宛てるクラスです。これはオプションとして、コンバータを通してこのソース・データを渡すことができます。

NLP プロセッサは `%iKnow.Source.Processor` クラスのサブクラスです。それぞれのプロセッサ・サブクラスは、ディレクトリからファイルを読み取る `%iKnow.Source.File.Processor` など、特定タイプのソースを読み取るように設計されています。

すべてのリスタは、そのリスタから取得したソースを処理することが可能な既定のプロセッサを備えています。既定では、そのリスタと同じパッケージ内のプロセッサ (Processor) というクラスを使用します。指定のリスタに対応するプロセッサがない場合、または一般的な `%iKnow.Source.Temp.Processor` を使用したい場合は、DefaultProcessor() メソッドをオーバーライドし、希望する既定のプロセッサを指定します。

22.1.4 リストの展開

ExpandList() メソッドは、インデックスを作成する必要があるすべてのソースをリストします。このメソッドは、カスタム・リスタのために特定タイプのソースの場所または構造全体をスキャンする方法を実装するユーザ定義サブクラスでオーバーライドする必要があります。このメソッドのパラメータは、対応する AddListToBatch() メソッドを呼び出す際に使用されるものと同じになります。パラメータは、実装するリスタによって異なります。lister.AddListToBatch() メソッドや loader.ProcessList() メソッドに指定するパラメータがユーザにわかるように、必ずリスタ固有の ExpandList() パラメータを文書化しておいてください。

ExpandList() のパラメータは以下のとおりです (指定順)。

- ・ Path : ソースが置かれる場所で、文字列で指定します。
- ・ Extensions : リストするソースを識別する 1 つまたは複数のファイル拡張子接尾語。文字列の %List として指定します。

- ・ Recursive : パスのサブディレクトリでソースを検索するかどうかを指定するブーリアン値。
- ・ Filter : リストするソースを制限するために使用するフィルタを指定する文字列。

詳細は、“プログラムによるテキスト・データのロード” の章の “[リスタ・パラメータ](#)” を参照してください。

22.2 カスタム・プロセッサ

プロセッサは、NLP 処理のために全ソースを一時グローバルにコピーするか、ソースの参照を一時グローバルに格納することができます。NLP エンジンはいずれの一時グローバルを使用して、テキストのインデックスを作成し、その結果を NLP グローバルに格納します。

リスタが対応するプロセッサを持たない場合は、`%iKnow.Source.Temp.Processor` が既定のプロセッサになります。これは各ソースの全テキストを一時グローバルにコピーします。提供される他のプロセッサは、ソースの参照を一時グローバルに格納します。`..StoreTemp` を使用すると、ソースのコピーを指定できます。または、`..StoreRef` を使用すると、ソースの参照の格納を指定できます。

22.2.1 メタデータ

リスタはソースをリストする一方で、ソースに追加する必要があるメタデータを抽出できます。リスタが提供するメタデータをシステムに知らせるために、関数 `..RegisterMetadataKeys(metaFieldNames)` を呼び出すことができます。metaFieldNames パラメータは、メタデータのキーと値のペアについてのキーを含む %List です。この後で、関数 `..SetMetadataValues(ref, metaValues)` を使用して、メタデータ値を提供できます。metaValues パラメータは、メタデータのキーと値のペアについての値を含む %List です。これらは、キーがリストされているのと同じ順序にする必要があります。

リスタにメタデータを設定したら、`GetMetadataKeys()` メソッドを実装することで、プロセッサでこのメタデータにアクセスできます。このメソッドは、メタデータのキーと値のペアについてのキーの %List を返します。これでプロセッサは、`FetchSource()` メソッドで `..SetCurrentMetadataValues(values)` を呼び出すために適切な値を設定できます。この values は、メタデータのキーと値のペアについての値の %List で、キーがレポートされたのと同じ順序になります。

22.3 カスタム・コンバータ

コンバータはソース・テキストからタグを除去することで、ソース・テキストをプレーン・テキストに変換します。タグとは、表示や印刷用にテキストをフォーマットするために使用される、コンテンツ以外の要素です。例えば、RTF (Microsoft リッチ・テキスト形式) ファイルからタグを除去したり、PDF ファイルからプレーン・テキストを抽出するために、コンバータを使用できます。コンバータはリスタによって呼び出され、ソース・テキストのインデックスを作成する前に適用されます。ソース・ドキュメントの形式によっては、ソース・コンバータの使用はオプションになります。

NLP には、サブクラス `%iKnow.Source.Converter.Html` というサンプル・コンバータが 1 つ用意されており、これを使用してソース・テキストから HTML タグを削除できます。これは、基本的な HTML コンバータで、HTML ソース・テキストの完全な変換をサポートするには、このコンバータのインスタンスをカスタマイズすることが必要な場合があります。

カスタム・コンバータを実装するには、ベース・コンバータ・クラス `%iKnow.Source.Converter` のいくつかのメソッドをオーバーライドする必要があります。

22.3.1 %OnNew

ユーザが提供する `%OnNew()` [コールバック・メソッド](#) は、`%New()` メソッドによって呼び出されます。これは、コンバータが必要とするパラメータの %List をパラメータとして使用します。

22.3.2 BufferString

BufferString() メソッドは、ドキュメント全体をコンバータにバッファするために、必要な回数だけ呼び出されます。それぞれの呼び出しは、data パラメータによってテキストのチャンクを提供します (最大 32K)。バッファするデータがなくなると、Convert() メソッドが呼び出されます。

22.3.3 Convert

Convert() メソッドは、バッファされたコンテンツを処理してデータをプレーン・テキストに変換したり (例 : RTF ファイル変換)、必要なデータをバッファから抽出します (例 : xml からのノード抽出)。変換されたデータは 32K より大きくなる可能性があるため、変換または抽出されたデータはバッファすることが必要になります。

22.3.4 NextConvertedPart

NextConvertedPart() メソッドは、Convert() メソッドの後で呼び出されます。このメソッドは 32K のチャンクで変換されたデータを返す必要があります。このメソッドが呼び出されるたびに、次のチャンクを返す必要があります。データがなくなると、このメソッドは空の文字列 ("") を返して、変換されたデータの抽出が終了したことを示します。

23

言語の識別

この章では、文のレベルで適用される自動言語識別 (ALI) の構成方法および使用方法を説明します。また、いくつかの言語固有の問題についても説明します。

23.1 自動言語識別の構成

NLP 構成により、ソース・ドキュメント・コンテンツのための言語環境が確立します。構成はどの特定のソース・データ・セットにも依存しません。構成は定義することができますが、既定の構成を採用することもできます。構成を指定しない場合、既定は英語専用となり、自動言語識別は行われません。

構成では以下の言語オプションを定義します。

- ・ ソース・ドキュメントに含まれる言語、つまり、テストする言語と適用する言語モデルを指定します。使用可能なオプションは、Czech/cs (チェコ語)、Dutch/nl (オランダ語)、English/en (英語)、French/fr (フランス語)、German/de (ドイツ語)、Japanese/ja (日本語)、Portuguese/pt (ポルトガル語)、Russian/ru (ロシア語)、Spanish/es (スペイン語)、Swedish/sv (スウェーデン語)、および Ukrainian/uk (ウクライナ語) です。2 文字の ISO 言語コードを使用して、言語を指定します。複数の言語を InterSystems IRIS リスト構造として指定できます。
- ・ 複数の言語を指定するときは、ブーリアン値を指定して自動言語識別を有効にします。

以下の例では、全ソース・テキストが英語またはフランス語であると想定した構成を作成し、自動言語識別をサポートしています。

ObjectScript

```
SET myconfig="EnglishFrench"
IF ##class(%iKnow.Configuration).Exists(myconfig) {
    SET cfg=##class(%iKnow.Configuration).Open(myconfig)
    WRITE "Opened existing configuration ",myconfig,!
}
ELSE {
    SET cfg=##class(%iKnow.Configuration).%New(myconfig,1,$LISTBUILD("en","fr"),",",1)
    DO cfg.%Save()
    IF ##class(%iKnow.Configuration).Exists(myconfig)
        {WRITE "Configuration ",myconfig," now exists",! }
    ELSE {WRITE "Configuration creation error" QUIT }
}
SET cfgId=cfg.Id
WRITE "with configuration ID ",cfgId,!
SET rnd=$RANDOM(2)
IF rnd {
    SET stat=##class(%iKnow.Configuration).%DeleteId(cfgId)
    IF stat {WRITE "Deleted the ",myconfig," configuration" }
}
ELSE {WRITE "No delete this time",! }
```

23.2 自動言語識別の使用

NLPは文ごとに自動言語識別を実行します。現在の構成で自動言語識別を有効にすると、NLPは各ソース・テキストの各文をテストして、構成で指定されたどの言語がその文で使用されているかを判別します。この識別は統計的確率です。これは、以下のような影響があります。

- ・ 構成で指定された複数言語のテキストが 1 つの文に含まれている場合、NLP は、その文で優勢な言語と判別された言語に文を割り当てます。
- ・ 構成で指定されていない言語（または NLP がサポートしていない言語）で文が記述されている場合、NLP は指定された構成言語の 1 つに文を割り当てます。

NLP はその後、この言語判別結果を使用して、CRC などの NLP 分析での判定を行います。

このように、ソース・テキストとソース・テキスト内の文は、異なる言語で記述される可能性もあります。NLP は、適用する言語モデルを自動的に判別します。また、自動言語識別は、その言語識別の信頼度を、パーセンテージを示す整数値で割り当てます。信頼度の範囲は 100 (完全に信頼できる) ~ 0 (不確定) となります。自動言語識別が有効でない場合、すべての文には信頼度 0 が割り当てられます。

23.2.1 言語識別クエリ

このドキュメント内の例で使用されているコーディングおよびデータの詳細は、“[サンプル・プログラムに関するメモ](#)”を参照してください。

以下の例では、GetTopLanguage() を使用して、ソースの言語および言語認識における信頼度を認識しています。言語認識は文レベルで実行されるので、ソースの言語はコンポーネントの文に対する言語認識の信頼度を平均化した結果となります。このメソッドは 2 文字の省略形で言語を返します（この場合、“en”）。totlangconf（文に対する言語信頼度の合計）は、numsent ではなく、numlangsnt で除算する必要があることに注意してください。これら 2 つの文の数は通常同じですが、常に同じではありません。これは、ソースに判別不能な言語の文が含まれる可能性があるためです。

ObjectScript

```
Configuration
SET myconfig="EnFr"
IF ##class(%iKnow.Configuration).Exists(myconfig)
{SET cfg=##class(%iKnow.Configuration).Open(myconfig) }
ELSE {SET cfg=##class(%iKnow.Configuration).%New(myconfig,1,$LISTBUILD("en","fr"),",",1)
DO cfg.%Save() }
SET cfgId=cfg.Id
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
SET domoref=##class(%iKnow.Domain).%New(dname)
DO domoref.%Save()
WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
GOTO ListerAndLoader }
ELSE { WRITE "DropData error ", $System.Status.DisplayError(stat)
QUIT}
ListerAndLoader
SET domId=domoref.Id
SET flister=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET stat=flister.SetConfig(myconfig)
IF stat '= 1 { WRITE "SetConfig error ", $System.Status.DisplayError(stat)
QUIT }
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT Top 10 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
```

```

SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=fliester.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
GetSources
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId)
SET i=1
WHILE $DATA(result(i)) {
  SET intId = $LISTGET(result(i),1)
  SET extId = $LISTGET(result(i),2)
  SET numsent = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
  WRITE !,extId," has ",numsent," sentences",!
  SET srclang = ##class(%iKnow.Queries.SourceAPI).GetTopLanguage(domId,intId,.totlangconf,.numlangsnt)

  WRITE "Source language is ",srclang,!,"with a confidence % of ",totlangconf/numlangsnt,!
  SET i=i+1
}

```

以下の例では、GetLanguage()を使用して、ソースの各文の言語および言語認識における信頼度を認識しています。このメソッドは、2 文字の省略形で言語を返し(ここでは“en”)、0 ~ 100 のパーセンテージで信頼度を返します。信頼度が 100% になることは、めったにありません。

ObjectScript

```

Configuration
SET myconfig="EnFr"
IF ##class(%iKnow.Configuration).Exists(myconfig)
{SET cfg=##class(%iKnow.Configuration).Open(myconfig) }
ELSE {SET cfg=##class(%iKnow.Configuration).%New(myconfig,1,$LISTBUILD("en","fr"),",",1)
      DO cfg.%Save() }
SET cfgId=cfg.Id
DomainCreateOrOpen
SET dname="mydomain"
IF (##class(%iKnow.Domain).NameIndexExists(dname))
{ WRITE "The ",dname," domain already exists",!
  SET domoref=##class(%iKnow.Domain).NameIndexOpen(dname)
  GOTO DeleteOldData }
ELSE
{ WRITE "The ",dname," domain does not exist",!
  SET domoref=##class(%iKnow.Domain).%New(dname)
  DO domoref.%Save()
  WRITE "Created the ",dname," domain with domain ID ",domoref.Id,!
  GOTO ListerAndLoader }
DeleteOldData
SET stat=domoref.DropData()
IF stat { WRITE "Deleted the data from the ",dname," domain",!!
         GOTO ListerAndLoader }
ELSE { WRITE "DropData error ",$System.Status.DisplayError(stat)
      QUIT}
ListerAndLoader
SET domId=domoref.Id
SET fliester=##class(%iKnow.Source.SQL.Lister).%New(domId)
SET stat=fliester.SetConfig(myconfig)
IF stat '= 1 { WRITE "SetConfig error ",$System.Status.DisplayError(stat)
              QUIT }
SET myloader=##class(%iKnow.Source.Loader).%New(domId)
QueryBuild
SET myquery="SELECT Top 10 ID AS UniqueVal,Type,NarrativeFull FROM Aviation.Event"
SET idfld="UniqueVal"
SET grpfld="Type"
SET dataflds=$LB("NarrativeFull")
UseLister
SET stat=fliester.AddListToBatch(myquery,idfld,grpfld,dataflds)
IF stat '= 1 {WRITE "The lister failed: ",$System.Status.DisplayError(stat) QUIT }
UseLoader
SET stat=myloader.ProcessBatch()
IF stat '= 1 {WRITE "The loader failed: ",$System.Status.DisplayError(stat) QUIT }
GetOneSource
DO ##class(%iKnow.Queries.SourceAPI).GetByDomain(.result,domId)
FOR i=1:1:10 {
  IF $DATA(result(i)) {
    SET intId = $LISTGET(result(i),1)
    SET extId = $LISTGET(result(i),2)
    SET myconf=0
    SET numSentS = ##class(%iKnow.Queries.SentenceAPI).GetCountBySource(domId,result(i))
    WRITE !,extId," has ",numSentS," sentences",!
  }
}

```

```

GetSentencesInSource
  SET sentStat=##class(%iKnow.Queries.SentenceAPI).GetBySource(.sent,domId,intId)
  IF sentStat=1 {
    SET i=1
    WHILE $DATA(sent(i)) {
      SET sentnum=$LISTGET(sent(i),1)
      WRITE "sentence:",sentnum
      SET lang = ##class(%iKnow.Queries.SentenceAPI).GetLanguage(domId,sentnum,.myconf)
      WRITE " language:",lang," confidence:",myconf,!
      SET i=i+1
    }
  }
}
ELSE { WRITE !,"That's all folks!" }
}

```

23.3 自動言語識別のオーバーライド

LanguageFieldName ドメイン・パラメータを使用して、自動言語識別をオーバーライドすることができます。有効時には、このパラメータは、各ソースのメタデータ・フィールドにアクセスすることにより、どの言語を適用するか判断します。メタデータ・フィールドには、ISO 言語コードが含まれます。メタデータ・フィールド・データが存在する場合、自動言語識別がそのソースに対してオーバーライドされます。メタデータ・フィールドが空白または無効である場合、自動言語識別がそのソースに対して使用されます。LanguageFieldName ドメイン・パラメータは、既定で無効となっています。詳細は、このドキュメントの付録の [“ドメイン・パラメータ”](#) を参照してください。

23.4 言語固有の問題

ドイツ語：ドイツ語の eszett (“ß”) の文字は、“ss” として[標準化](#)されます。ドイツ語では通常、EnableNgrams [ドメイン・パラメータ](#)を設定する必要があります。

A

ドメイン・パラメータ

この付録には、使用可能なドメイン・パラメータがリストされています。ドメイン・パラメータ名は、大文字と小文字が区別されます。各ドメイン・パラメータには、%IKPublic マクロ相当の機能 (\$\$\$IKPFULLMATCHONLY など) が用意されています。ドメイン・パラメータは、そのパラメータ名ではなく、マクロ相当の機能によって指定するプログラミング方法をお勧めします。ドメイン・パラメータの設定の詳細は、“[NLPドメインの定義](#)”を参照してください。

ドメイン・パラメータは、特定のドメインに対してまたは“システム全体”に定義できます。システム全体とは、現在のネームスペースの現在のドメインおよび将来のドメインすべてに対して定義するということです。

ドメイン・パラメータは、Basic と Advanced の 2 つのグループに分かれています。Basic のパラメータは、NLP の既定の動作をカスタマイズする場合に役に立ちます。Advanced のパラメータは、NLP の動作とパフォーマンスを大幅に変更するため、注意して使用する必要があります。

テーブル I-1: Basic ドメイン・パラメータ

パラメータ	説明
-------	----

パラメータ	説明
DefaultConfig	<p>\$\$\$IKPDEFAULTCONFIG : ドメインに使用される NLP 構成 の名前を指定する文字列。既定では、構成はドメインに割り当てられてなく、文字列 DEFAULT を返すことによってこれが指定されます。現在のネームスペースで定義されたすべて構成の名前を指定できます。構成をこのパラメータに割り当てる場合は、構成が存在する必要があります。構成が存在しない場合は、このパラメータは変更されないままになります。SetConfig() または ProcessBatch() によって、DefaultConfig 値をオーバーライドできます。</p>
EnableNgrams	<p>\$\$\$IKPENABLENGRAMS : ブーリアン・パラメータ。1 に設定すると、NLP はドメインの N-gram を生成します。N-gram は単語内の 類似エンティティ・マッチング に使用されます。0 に設定すると、NLP は、パーツのみ、単語全体、または単語の先頭文字をマッチングします。既定値は 0 です。ドメイン・レベルでは、空のドメイン (NLP データをまだ含まないドメイン) の EnableNgrams 設定のみを変更できます。システム全体のレベルでは、現在のネームスペースにテキスト・データを含むドメインが存在しない場合、EnableNgrams 設定のみを変更できます。N-gram マッチングを使用すると、NLP によって格納されるデータのサイズが大幅に増大し、パフォーマンスに大きな影響を与える可能性があります。したがって、これは必要な場合のみ有効にしてください。ほとんどの言語では、N-gram マッチングを使用するべきではありません。ただし、ドイツ語テキストのマッチング処理では、しばしば N-gram マッチングが必要になります。</p>
IgnoreDuplicateExtIds	<p>\$\$\$IKPIGNOREDUPLICATEEXTIDS : ブーリアン・パラメータ。1 に設定すると、NLP では、既にロードされたソースと同じ外部 ID を持つソースがロードされた場合にエラーの記録が行われません。最後のロード以降に追加された新しいソースを追加するために、以前にロードされた場所からソースをロードする場合は、1 に設定することをお勧めします。既定値は 0 です。</p>
IgnoreEmptyBatch	<p>\$\$\$IKPIGNOREEMPTYBATCH : ブーリアン・パラメータ。0 に設定すると、ローダでは、ソースを指定しないバッチ・ロードが指定された場合に \$\$\$IKNothingToProcess エラーが発行されます。1 に設定すると、ローダではこのエラーが生成されません。既定値は 0 です。</p>

パラメータ	説明
MAT:DefaultProfile	<p>\$\$\$IKPMATDEFAULTPROFILE : このパラメータでは、ドメインの既定として作成が必要なユーザ定義マッチング・プロファイルの名前が取得されます。マッチング・プロファイルがこのパラメータに割り当てる場合は、マッチング・プロファイルが存在する必要があります。ユーザ定義マッチング・プロファイルが（ドメイン固有ではなく）ネームスペース全体として定義されている場合、名前の前にゼロ・コロン (0:) を追加する必要があります。例えば、"0:NoDomainProfile" となります。このパラメータを設定しない場合は、ドメインの既定として NLP の既定のマッチング・プロファイルが使用されます。詳細は、“スマート・マッチング：ディクショナリの使用”の章の“マッチング・プロファイルの定義”を参照してください。</p> <p>MatchSource() メソッドまたは MatchSources() メソッドにカスタム・マッチング・プロファイルを指定することで、ドメイン既定のマッチング・プロファイルをオーバーライドできます。</p>
MAT:SkipRelations	<p>\$\$\$IKPMATSKIPRELATIONS : ディクショナリ・マッチング用のブーリアン・パラメータ。値が 1 (既定) の場合は、関係ではなく、概念のみが指定され、エンティティ・マッチングの間にマッチングされます。(関係は、CRC およびパスのマッチング処理間にマッチングされます。)関係エンティティ・マッチングのスキップによってパフォーマンスを大幅に強化できます。値が 0 の場合は、関係エンティティ・マッチングが実行されます。ディクショナリに単一エンティティ用語である関係が含まれている場合のみ、このパラメータに 0 を設定する必要があります。</p>

パラメータ	説明
SortField	<p>\$\$\$IKPSORTFIELD : ブーリアン・パラメータ。すべての NLP エンティティには、頻度 (全ソースにおけるエンティティの出現回数) と分散 (エンティティが出現するソースの数) という 2 つの整数カウントが関連付けられています。0 に設定すると、NLP の既定では頻度によって並べ替えられます (\$\$\$SORTBYFREQUENCY)。1 に設定すると、NLP の既定では分散によって並べ替えられます (\$\$\$SORTBYSPREAD)。既定値は 0 です。ドメイン・レベルでは、空のドメイン (NLP データをまだ含まないドメイン) の SortField 設定のみを変更できます。システム全体のレベルでは、現在のネームスペースにテキスト・データを含むドメインが存在しない場合、SortField 設定のみを変更できます。</p> <p>または、このドメイン全体の並べ替え順序は、2 番目のパラメータ (sortField) を %iKnow.Domain クラスの Create() または GetOrCreateId() メソッドに指定することで変更できます。0 = 頻度による並べ替え (既定)、1 = 分散による並べ替えとなります。並べ替え順序の既定値は、必要な場合のみ変更してください。並べ替え順序の既定値は、空のドメイン (まだ NLP データを含んでいないドメイン) に対してのみ変更できます。</p> <p>特定の個々のクエリでは、並べ替え順序を指定 (\$\$\$SORTBYFREQUENCY または \$\$\$SORTBYSPREAD) するか、あるいは現在のドメインの既定値 (\$\$\$SORTBYDOMAINDEFAULT) を使用できます。</p>
Status	<p>\$\$\$IKPSTATUS : ブーリアン・パラメータ。1 に設定すると、NLP はソースをロードするプロセスの進行状況について詳細なステータス情報を表示します。0 に設定すると、NLP ではこの情報が表示されません。既定値は 0 です。</p>
Stemming	<p>\$\$\$IKPSTEMMING : ブーリアン・パラメータ。1 に設定すると、NLP はドメインで語幹解析を有効にします。0 に設定すると、NLP は語幹解析を実行しません。既定値は 0 です。</p>

テーブル I-2: Advanced ドメイン・パラメータ

パラメータ	説明
-------	----

パラメータ	説明
EntityLevelMatchOnly	<p>\$\$\$IKPENTITYLEVELMATCHONLY : ディクショナリに対するマッチング時に実行されるマッチング処理タイプの制限に使用するブーリアン・パラメータ。既定では、NLP はエンティティ、CRC、パス、および文をマッチングします。このパラメータを設定して、マッチングをエンティティのみに制限できます。この結果、一致結果がはるかに多くなる可能性があります。既定値は 0 です。このパラメータを変更すると、このドメイン以降のすべてのマッチング処理に影響が及びます。したがって、このパラメータを変更する前にマッチング済みのソースについては、この変更を反映するために明示的に再度マッチングさせる必要があります。</p>
FullMatchOnly	<p>\$\$\$IKPFULLMATCHONLY : ディクショナリに対するマッチング時に実行されるマッチング処理タイプの制限に使用するブーリアン・パラメータ。このパラメータを設定して、マッチングを完全一致のみに制限できます。このオプションを (1) に設定すると、部分一致と不規則一致は無視されます。既定値は 0 です。このパラメータを変更すると、このドメイン以降のすべてのマッチング処理に影響が及びます。したがって、このパラメータを変更する前にマッチング済みのソースについては、この変更を反映するために明示的に再度マッチングさせる必要があります。</p>
LanguageFieldName	<p>\$\$\$IKPLANGUAGEFIELDNAME : メタデータ・フィールドの名前を指定する文字列です。ソースのロード時に生成された既存のメタデータ・フィールドが設定された場合、NLP はそのメタデータ・フィールドの値 (設定されている場合) を、対応するソースの処理時に使用するための言語として使用します。このオプションは、自動言語識別をオーバーライドします。メタデータ・フィールドの値は、現在の構成オブジェクトで指定した言語用の 2 文字の ISO 言語コード (\$\$\$IKLANGUAGES 参照) とする必要があります。</p>

パラメータ	説明
MAT:StandardizedForm	<p>\$\$\$IKPMATSTANDARDIZEDFORM : ドメインの標準化形式マッチングを有効にするには、このドメイン・パラメータを目的の標準化関数に対して、SET</p> <pre>stat=domain.SetParameter("MAT:StandardizedForm", "%Text")</pre> <p>のように設定します。標準化形式マッチングでは、ディクショナリ用語の標準化形式を使用した、単数形/複数形または動詞形などのソース・テキスト内の同一単語の別形式と単一ディクショナリ用語とのマッチングをサポートしています。そのようなドメインで作成されたディクショナリ用語はいずれも、“標準化”ディクショナリ要素となり、ソース内エンティティの標準化形式と比較した一致回数になります。正しい標準化アルゴリズムを使用するためには、ディクショナリ用語が正しい言語アノテーション (CreateDictionary() メソッドのパラメータ) で作成され、かつソースに適切な言語が (構成における適切な言語モデルの選択または 自動言語識別 の使用のいずれかにより) 割り当てられることが大切です。</p>
MetadataAPI	<p>\$\$\$IKPMETADATAAPI : %iKnow.Queries.MetadataAPI の拡張に使用するメタデータ API クラスを指定する整数。既定は %iKnow.Queries.MetadataAPI です。空のドメイン (NLP データをまだ含まないドメイン) に対してのみ、MetadataAPI 設定を変更できます。</p>
QUERY:MinTopConceptLength	<p>\$\$\$IKPMINTOPCONCEPTLENGTH : GetTop() クエリが返すことができる最小の概念 (最小の文字数) を指定する整数。このパラメータは、GetTop() の結果から無意味に短い概念をフィルタするために使用されます。既定値は 3 で、長さが 3 文字以上の概念を GetTop() で返すように指定します。この最小の文字カウントには、概念の単語や句読記号間のスペースが含まれます。</p>
SimpleExtIds	<p>\$\$\$IKPSIMPLEEXTIDS : ソースの外部 ID の形式を指定するために使用するブーリアン・パラメータ。0 に設定すると、NLP では、外部 ID として完全参照を格納します。1 に設定すると、NLP では、外部 ID としてローカル参照を格納します。既定値は 0 です。空のドメイン (NLP データをまだ含まないドメイン) に対してのみ、SimpleExtIds 設定を変更できます。</p>
SkipExtIdCheck	<p>\$\$\$IKPSKIPEXTIDCHECK : 重複する外部 ID をチェックするかどうかを指定するブーリアン・パラメータ。1 に設定すると、NLP では、ソースのロード時に、重複する外部 ID が既に存在するかどうかのチェックをスキップします。0 に設定すると、NLP では重複する外部 ID をチェックします。既定値は 0 です。</p>

パラメータ	説明
UseEntityVectorsJP	<p>\$\$\$IKPUSEENTITYVECTORSJP: 文が日本語である場合に、ドメインでエンティティ・ベクトル・アルゴリズムを使用してパスを解析して保存するかどうかを指定するブーリアン・パラメータ。既定では、日本語のテキストに対するエンティティ・ベクトルの使用が有効になっています。これは、“コンセプトの概要”の“パス”で説明するように、ほとんどの場合に推奨できる動作です。</p> <p>UseEntityVectorsJP を 0 に設定していると、西洋語のテキストでパスの特定に使用する場合と同じアルゴリズムが NLP で使用されて、日本文が分析されます。</p> <p>ドメイン・レベルでは、空のドメイン (NLP データをまだ追加していないドメイン) の UseEntityVectorsJP パラメータのみを変更できます。システム全体のレベルでは、テキスト・データを持たないドメインが現在のネームスペースに存在しない場合に、UseEntityVectorsJP パラメータのみを変更できます。</p>

