119270, Москва, Лужнецкая наб., д. 6,
стр.1, офис 214, ООО «ЭР СИ О»
Тел./факс: (495) 287-98-87
E-mail: info@rco.ru
http://www.rco.ru

Руководство разработчика
RCO Categorization Engine – библиотека категоризации текстов

Версия 2.0

(Microsoft Windows, Unix)

Москва, 2014

В содержание данного документа могут быть внесены изменения без предварительного уведомления. Названия организаций, имена и даты, используемые в качестве примеров, являются вымышленными, если не оговорено обратное.

Содержание

Обзор

Библиотека категоризации текстов позволяет решать следующие задачи:

- На основании лексических профилей эффективно определять принадлежность текста к заданному множеству категорий;
- Для каждого термина из лексических профилей, обнаруженного в тексте, получить количество его вхождений в текст, а также позиции терминов в тексте.

Основными областями применения библиотеки являются:

- Тематическая категоризация текстов в электронных библиотеках, информационно-поисковых и информационно-аналитических системах;
- Тематический таргетинг в баннерных сетях;
- Мониторинг ключевых слов и словосочетаний в системах мониторинга и сбора информации.

К отличительным возможностям библиотеки следует отнести:

- Возможности по очистке web-страниц:
  - Автоматическая коррекция кодовой страницы русского языка;
  - Удаление навигационных элементов сайта, загромождающих страницу.
- Гибкие настройки идентификации терминов в тексте, а именно:
  - В точности, как написано в профиле;
  - С учетом всех словоформ при помощи морфоанализатора русского языка;
  - Явно задав все требуемые к отождествлению словоформы.
- Задание терминов в виде поисковых выражений с использованием следующих операторов:
  - Слова, словосочетания;
  - Задание расстояния межу словами;
  - Логические операторы – «И», «ИЛИ», «И НЕ».
- Специальная обработка отдельных терминов:
  - Термин обязательно должен встретиться для отнесения текста к категории;
  - Термин не должен встретиться для отнесения текста к категории.

В библиотеке реализована векторная модель категоризации, которая включает в себя четыре настраиваемых компонента:

- Весовые коэффициенты терминов профиля (значения задаются пользователем);
- Весовые коэффициенты терминов документа (способ вычисления задается пользователем: бинарные, частотные);
- Нормирующий множитель (способ вычисления задается пользователем: евклидова норма, длина текста);
- Пороговое значение (задается пользователем).

В процессе категоризации *степень соответствия текста профилю* рассчитывается как сумма произведений *весовых коэффициентов терминов профиля* и *весовых коэффициентов терминов документа*, деленая на *нормирующий множитель*. Результат сравнивается с *пороговым значением*. В случае его превышения документ считается принадлежащим рубрике.

При необходимости вне библиотеки разработчиком могут быть реализованы и другие методы категоризации текстов.

Морфоанализатор, включенный в библиотеку, имеет следующие характеристики:

- Реализованы следующие методы анализа:
  - Точный анализ известных слов по словарю объемом более 115 тысяч слов, что покрывает более 3-х миллионов словоформ;
  - Высоко достоверный анализ неизвестного слова на основе комплекса правил словообразования и словоизменения;
  - Вероятностный анализ посредством соотнесения с моделями словоизменения часто встречающихся слов на основе оценки флективной и суффиксальной частей слова.
  - Объем бинарного словаря – 3 Мб;
- Скорость морфологического анализа – более 100 тысяч слов в секунду (процессор **AMD Athlon**, 1000 МГц).

Кодовая страница, используемая при работе библиотеки, – **Win1251**.

В состав библиотеки входят следующие файлы:

gpcat.h	файл заголовка с описанием констант и прототипов функций
gpcat.lib	lib файл для компоновки в среде **MS Visual C++ 2005** (только в **Windows**-версии)
gpcat.dll	динамическая библиотека (только в **Windows**-версии)
libgpcat.so	динамическая библиотека (только в **Unix**-версии)
morphdct.dat	бинарный морфологический словарь
casample.cpp	исходный код тестового примера на **C++**

Отличия от предыдущих версий

Ниже перечислены отличия данной версии библиотеки от предыдущих.

Отличия от версии 1.1

- Введена возможность отождествления словосочетаний с учетом расстояния. Если за последним термином словосочетания следует выражение /N, то оно интерпретируется как оператор для учета расстояния. В этом случае словосочетание ищется в тексте как последовательность слов длины не более N, в которой должны встретиться слова, составляющие словосочетание, и в том порядке, как они указаны в словосочетании.
- Добавлена возможность использования морфологического словаря (*CA_P_STR_GPMORPH_DICTIONARY*). Термины, для которых явно не указана парадигма и не задан модификатор для отождествления «как есть», идентифицируются в тексте с учетом словоформ русского языка при помощи встроенного морфоанализатора.

Отличия от версии 1.0

- Добавлена расширенная диагностика ошибок загрузки лексических профилей:

CA_Q_ERR_LOG_COUNT	Число записей об ошибках, обнаруженных при загрузке лексического профиля.
CA_Q_ERR_LOG_STR	Текст записи об ошибке, обнаруженной при загрузке лексического профиля.

- Добавлен параметр, влияющий на обработку ошибок в лексических профилях:

CA_P_BOOL_IGNORE_INCORRECT_PARADIGMS	Параметр определяет, игнорировать ли некорректно составленные парадигмы. Число слов в каждом варианте употребления словосочетания в парадигме должно совпадать. При нарушении этого ограничения профиль считается некорректным. При указании данного параметра неверно составленная парадигма игнорируется, и загрузка профиля продолжается.

Планы на следующие версии

- Введение нового типа терминов – многозначные. Если в тексте встретились только многозначные термины из профиля, то даже в случае формального превышения суммой вкладов встретившихся терминов порогового значения, текст не будет отнесен к профилю. Смысл введения нового типа – многозначные термины требуют подтверждения обычными терминами или обязательными.
- Поддержка параллельной обработки текста в контексте одного экземпляра загруженных профилей. На текущий момент реализация библиотеки требует заводить разные дескрипторы для параллельной обработки текстов.

Принципы работы

Настоящий раздел покрывает следующие темы:

Основные определения

Весовой коэффициент термина профиля – численная величина, присваиваемая термину при создании лексического профиля. Используется при вычислении соответствия текста профилю. Чем выше значение весового коэффициента, тем более значим термин для темы, описываемой профилем. Весовой коэффициент может принимать и отрицательные значения.

Весовой коэффициент термина документа – величина, вычисляемая при сравнении текста с профилем и характеризующая значимость термина в тексте. В библиотеке реализовано два способа расчета данного весового коэффициента:

- Бинарный – термину присваивается вес 1, если термин встретился, либо – 0 в противном случае;
- Частотный – термину присваивается вес, равный количеству раз, которое термин встретился в документе.

Вклад термина в степень соответствия текста профилю – в реализации библиотеки это произведение: *<весовой коэффициент термина в профиле> * <весовой коэффициент термина в документе> * <нормирующий множитель>*.

Категоризация – установление соответствия текста заданной теме (перечню тем), заключающееся в идентификации терминов лексических профилей текста, проверке условий для обязательных и исключающих терминов, вычислении степени соответствия текста профилям, а также принятие решения об отнесении текста к темам.

Лексический профиль – совокупность данных, описывающих заданную тему. Включает в себя перечень терминов, весовые коэффициенты, пороговое значение, указание на способ сравнения профиля с текстом. Используется для категоризации текстов.

Нормирующий множитель – численная величина, позволяющая привести возможные значения степени соответствия текстов профилю к единой шкале, почти не зависящей от размера текста (при попытке категоризации ***«Войны и мира»*** в тексте встретятся термины практически любого профиля с высокими частотами). В библиотеке реализовано два способа расчета нормирующих множителей:

- по длине текста – общее число слов текста;
- по длине вектора, описывающего документ в терминах профиля. Используется евклидова норма – корень из суммы квадратов *весовых коэффициентов терминов документа*.

Нормировка по длине вектора, как правило, применяется в методах машинного обучения для настройки весовых коэффициентов терминов профиля и порогового значения. Нормировка по длине текста применяется при ручной настройке профилей. Нормировка по длине текста обеспечивает несколько лучшее качество классификации при машинном обучении, но при этом усложняется дальнейшая ручная настройка профиля.

Парадигма – возможные варианты употребления слов, словосочетаний. Для отдельных слов – это все варианты их написания в соответствии с правилами склонения или спряжения в зависимости от части речи. Для словосочетаний при перечислении вариантов возможно использование правил согласования между словами по роду, числу, падежу.

Пороговое значение, порог – численная величина, которой должна достичь *степень соответствия текста профилю* для отнесения текста к теме.

Степень соответствия текста профилю – численная величина, показывающая, насколько хорошо текст соответствует профилю (теме). При принятии решения о соответствии текста профилю данная величина сравнивается с пороговым значением. В библиотеке данная величина вычисляется как сумма вкладов каждого термина.

Термин – логическое выражение, состоящее из слов и словосочетаний, возможно с указанием расстояния между словами, связанных логическими операторами «И», «ИЛИ», «И НЕ». С термином связаны *весовой коэффициент*, *тип термина*, а также, возможно, перечень *парадигм*, описывающих, идентифицируемые формы слов и словосочетаний, составляющих термин.

Тип термина – способ интерпретации факта наличия термина в тексте при принятии решения о соответствии текста профилю. В библиотеке поддерживаются три типа терминов:

- Обычный – может встретиться, а может – нет;
- Обязательный – должен встретиться (иначе текст не будет отнесен к профилю);
- Исключающий – не должен встретиться (в противном случае текст не будет отнесен к профилю);

Значения весовых коэффициентов обычных и обязательных терминов используются для вычисления степени близости текста к профилю. Значения весовых коэффициентов игнорируются.

Следующие определения даны для ознакомления читателя со смежным вопросом – как строить профили автоматически.

Извлечение терминов – выделение терминов из текста для использования их в дальнейшем в качестве классификационных признаков. Простейший случай – выделение слов при построении полнотекстового индекса. Более сложный – выделение слов и словосочетаний в результате ассоциативно-статистического либо синтактико-семантического анализа. Библиотека не включает в себя средств извлечения терминов.

Обучающее множество – набор текстов, являющихся положительными и отрицательными примерами для профиля.

Обучение – вычисление *весовых коэффициентов* и *порогового значения* на основе заданного набора терминов и наборов положительных и отрицательных примеров. Цель обучения – настройка *весовых коэффициентов* и *порогового значения* таким образом, чтобы процедура категоризации относила *положительные примеры* к теме, характеризуемой профилем, а *отрицательные* – нет. Т.е. чтобы степень соответствия всех *положительных примеров* превышала *пороговое значение*, а для *отрицательных* – была ниже. Алгоритмы обучения, как правило, ищут оптимум некоторой целевой функции (например, F-мера или сумма ошибок). Библиотека не включает в себя средств обучения.

Отбор терминов – отбор терминов для включения в лексический профиль. Для каждого извлеченного из текста *положительных примеров* термина проверяется ряд критериев, на основе которых принимается решение о включении в профиль. В частности, используются такие характеристики, как *полнота* и *точность* описания термином *тестового множества*. Как правило, требуется, чтобы полнота и точность были не менее заданных констант. Библиотека не включает в себя средств отбора терминов.

Положительный/отрицательный пример – текст, о котором заранее известно, что он относится или не относится к теме. Если текст относится к теме, то он называется *положительным примером*, иначе – *отрицательным примером*. При использовании методов машинного обучения с каждым профилем всегда связаны наборы положительных и отрицательных примеров. *Положительные примеры* используются для автоматического *извлечения терминов*, характеризующих тематический профиль. Положительные и отрицательные примеры в совокупности используются для обучения – вычисления *весовых коэффициентов* и *порогового значения*.

Полнота описания термином обучающего множества – отношение количества *положительных примеров*, в которых встретился *термин*, к общему числу *положительных примеров*.

Полнота результатов категоризации – отношение количества *положительных примеров, степень соответствия* которых превышает *пороговое значение*, к общему числу *положительных примеров*.

Тестовое множество, тестовая подборка – набор текстов, используемый для проверки качества построенного профиля на текстах, не входящих в *обучающее множество*.

Точность описания термином обучающего множества – отношение количества *положительных примеров*, в которых встретился *термин*, к общему числу примеров, в которых встретился *термин*.

Точность результатов классификации – отношение количества *положительных примеров, степень соответствия* которых превышает *пороговое значение*, к общему числу примеров, *степень соответствия* которых превышает *пороговое значение* (среди них могут быть и *отрицательные*).

Порядок работы с библиотекой

Перед началом работы с библиотекой необходимо создать дескриптор процедуры категоризации при помощи функции **`CAInitialize`**.

Затем следует установить параметры процедуры при помощи вызова **`CASetParameter`**. В текущей реализации библиотеки доступны такие параметры, как пороговое значение (*CA_P_DBL_LINK2TEXT_RATIO*) отношения ссылок к тексту в структурном блоке HTML-документа, при котором в случае очистки блок считается навигационным и исключается из обработки, путь к словарю морфологического анализа (*CA_P_STR_GPMORPH_DICTIONARY*) и прочие параметры.

Далее следует загрузить все лексические профили, последовательно обращаясь к функции **`CALoadProfile`**. При наличии ошибок или предупреждений об ошибках при загрузке профиля посредством функции **`CAQueryInfo`** с параметрами *CA_Q_ERR_LOG_COUNT* и *CA_Q_ERR_LOG_STR* можно получить все сообщения о нарушениях формата.

После загрузки профилей все готово для категоризации текстов. Категоризацию можно осуществлять либо вызовом **`CAProcessText`**, либо **`CAProcessTextEx`**. Различия в вызовах в том, что второй вызов позволяет соотнести текст с конкретным заданным профилем. В обоих вызовах можно выбрать опции предобработки – коррекцию кодовой страницы текста (*CA_OPT_DETECT_CODE_PAGE*), очистку HTML-страниц от навигационных элементов(*CA_OPT_CLEAN_PAGE*).

Получить информацию о результатах категоризации можно при помощи функции **`CAQueryInfo`** с различными параметрами. Например, при вызове с параметром *CA_Q_DOC_CATEGORY_COUNT* функция вернет число лексических профилей, к которым был отнесен текст.

По окончании работы с библиотекой требуется освободить дескриптор процедуры категоризации **`CAUninitialize`**.

Следует также отметить два общих соглашения:

- Функции библиотеки возвращают одно из следующих значений:

Таблица 1

CA_SUCCESS	Функция успешно выполнена.
CA_WARN_INVALID_FORMAT	Предупреждение о нарушении формата задания парадигмы.
CA_ERR_INVALID_FORMAT	Ошибка: нарушение формата профиля.
CA_ERR_INVALID_PARAMETER	Ошибка: в функцию переданы некорректные параметры.
CA_ERR_OUT_OF_MEMORY	Ошибка: недостаточно памяти для выполнения операции.
CA_ERR_PROFILE_ALREADY_EXISTS	Ошибка: профиль с данным ИД уже загружен.
CA_ERR_MORPHOLOGY_INITIALIZING	Ошибка: не удалось инициализировать модуль морфологии.
CA_ERR_MORPHOLOGY_REINITIALIZING	Ошибка: не удалось реинициализировать модуль морфологии.
CA_ERR_INTERNAL	Ошибка: непредвиденная ошибка при выполнении функции.

- При возврате в процессе работы с **`CAQueryInfo`** указателей на строки библиотека отвечает за освобождение зарезервированной под них памяти.

Интерфейс библиотеки

Описание интерфейса библиотеки состоит из двух разделов. В первом разделе приведены функции для управления процессом категоризации текстов. Во втором разделе приведено описание доступа к результатам категоризации.

В библиотеке доступны следующие функции:

CAInitialize

CAUninitialize

CAGetParameter

CASetParameter

CALoadProfile

CAProcessText

CAProcessTextEx

CAGetAnnotation

CAQueryInfo

CAInitialize

Функция **CAInitialize** используется для создания дескриптора процедуры категоризации и должна быть вызвана перед началом работы.

```
int CAInitialize (
   CA_HANDLE* pCA // дескриптор процедуры категоризации
);
```

Параметры

pCA
[вых] Указатель на дескриптор процедуры категоризации.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

> **Замечание.** Дескриптор по завершении работы должен быть освобожден посредством вызова **CAUninitialize**.

Пример вызова

```
CA_HANDLE hCA;
int nRes = CAInitialize( &hCA );
```

CAUninitialize

Функция **CAUninitialize** используется для освобождения ресурсов, связанных с дескриптором процедуры категоризации. Должна вызваться по окончании работы.

```
int CAUninitialize(
   CA_HANDLE hCA // дескриптор процедуры категоризации
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

> **Замечание.** Освобождаемый дескриптор должен быть создан при помощи вызова **CAInitialize**.

Пример вызова

```
int nRes = CAUninitialize( hCA );
```

См. также

CAInitialize

CAGetParameter

Функция **CAGetParamater** используется для получения значений параметров процедуры категоризации.

```
int CAGetParameter(
   CA_HANDLE hCA, // дескриптор
   unsigned int uParamType, // тип параметра
   void* pValue // указатель на переменную-приемник
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

uParamType
[вх] Тип запрашиваемого параметра. От него зависит тип переменной-источника.

CA_P_DBL_LINK2TEXT_RATIO	Значение порога отношения ссылок к тексту в структурном блоке HTML-документа, при котором в случае очистки блок считается навигационным и исключается из обработки. Значение по умолчанию 0,9.
CA_P_BOOL_IGNORE_INCORRECT_ PARADIGMS	Параметр определяет, игнорировать ли некорректно составленные парадигмы. Число слов в каждом варианте употребления словосочетания в парадигме должно совпадать. При нарушении этого ограничения профиль считается некорректным. В случае задания этого параметра неверно составленная парадигма игнорируется и загрузка профиля продолжается. Исходное значение 0.
CA_P_BOOL_OPTIMAL_COVERAGE	Возможны два способа отождествления терминов в тексте. Изначально все термины распознаются независимо с возможными перекрытиями. Возможен вариант отождествления, когда термины не должны пересекаться, и при этом вычисляется оптимальное покрытие текста терминами. **Пример**. Текст *ABCD*, термины *AB, BC, CD*. В первом случае сработают все термины, а во втором – только *AB* и *CD*, а *BC* не сработает, т.к. дает худшее покрытие. Значение по умолчанию *0*.
CA_P_STR_STRONG_DELIMITERS	Множество неалфавитно-цифровых символов, помимо пробельных, которые при разборе текста должны служить безусловными разделителями слов. Значение по умолчанию: '(',')','|','<','>','/','\'',0x84,0x91,0x92,0x93, 0x94,0xAB,0xBB,'{','}','[',']' В библиотеке также задано множество разделителей, которые разбивают слова в

	зависимости от контекста (если следом идет любой разделитель или пробельный символ): ';','-','.','!','?','\",',',':' В случае пересечения первое множество имеет приоритет над вторым. Если требуется изменить правила разбиения текста на слова, то можно некоторые контекстные разделители задать в списке безусловных разделителей. Например, можно разделять слова по символу “-”.
CA_P_INT_ANNOTATE_MAX_SENTENCE_COUNT	Максимальное количество предложений в аннотации.
CA_P_INT_ANNOTATE_MAX_TEXT_LENGTH	Максимальное количество символов в аннотации (без учета количества разделителей и разметки для подсветки слов).
CA_P_DBL_ANNOTATE_TERM_REOCCURRENCE_MULTIPLIER	Коэффициент взвешивания повторно встретившихся в предложении терминов при оценке веса предложения в аннотации.
CA_P_STR_ANNOTATE_FRAGMENT_DELIMITER	Строка разделителей фрагментов в тексте аннотации.
CA_P_STR_ANNOTATE_HIGHLIGHT_BEG	Строка начала подсветки терминов в тексте аннотации.
CA_P_STR_ANNOTATE_HIGHLIGHT_END	Строка окончания подсветки терминов в тексте аннотации.

pValue
[вх/вых] Указатель на переменную-приемник. Может иметь тип **char****, **int***, **double***.

Изменение значения **pValue* не допускается.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **`GetLastError`** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Пример вызова

```
double dblCleansing;
int nRes = CAGetParameter( hCA, CA_P_DBL_LINK2TEXT_RATIO,
   &dblCleansing );
```

См. также

`CAInitialize`, **`CALoadProfile`**

CASetParameter

Функция **CASetParamater** используется для установления значений параметров процедуры категоризации.

```
int CASetParameter(
   CA_HANDLE hCA,
// дескриптор
   unsigned int uParamType,
// тип параметра
   void* pValue
// указатель на переменную-источник
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

uParamType
[вх] Тип запрашиваемого параметра. От него зависит тип переменной-источника.

Полный перечень параметров приведен в описании функции **CAGetParameter**. Кроме того, для инициализации морфологических словарей используются еще и нижеперечисленные параметры:

CA_P_STR_GPMORPH_DICTIONARY	Путь к словарю морфологического анализа.
CA_P_STR_GPMORPH_EN_DICTIONARY	Путь к английскому словарю морфологического анализа.

pValue
[вх] Указатель на переменную-приемник. Может иметь тип **char***, **int***, **double***.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Пример вызова

```
double dblCleansing = 0.8;
int nRes = CASetParameter( hCA, CA_P_DBL_LINK2TEXT_RATIO,
   &dblCleansing );
```

См. также

CAInitialize, **CAGetParameter**

CALoadProfile

Функция **CALoadProfile** используется для загрузки лексического профиля с целью дальнейшего его сопоставления с текстами при вызове **CAProcessText** и **CAProcessTextExt**. Описание формата профиля приведено в Приложении 1.

```
int CALoadProfile(
   CA_HANDLE hCA,
// дескриптор
   char* pszProfile,
// буфер с профилем
   int* pProfileId,
// указатель на ИД профиля
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

pszProfile
[вх] Указатель на буфер с загружаемым профилем.

pProfileId
[вх] Указатель на переменную, что содержит идентификатор профиля. Если *pProfileId* равно 0, то идентификатор профиля считывается из содержимого профиля.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания

В случае ошибок или предупреждений об ошибках при загрузке профиля при помощи функции **CAQueryInfo** с параметрами *CA_Q_ERR_LOG_COUNT* и *CA_Q_ERR_LOG_STR* можно получить все сообщения об обнаруженных нарушениях формата.

На работу функции также влияет параметр процедуры категоризации *CA_P_BOOL_IGNORE_INCORRECT_PARADIGMS*, устанавливаемый вызовом **CASetParameter** (в случае задания данного параметра неверно составленная парадигма игнорируется и загрузка профиля продолжается).

Пример вызова

```
int nRes = CALoadProfile( hCA, pszProfile, 0 );
```

См. также

CAInitialize, **CASetParameter**, **CAQueryInfo**

CAProcessText

Функция **CAProcessText** используется для категоризации текста. Функция **CAQueryInfo**, вызванная с различными параметрами, делает эти результаты доступными.

```
int CAProcessText(
   CA_HANDLE hCA, // дескриптор
   char* pszHtml, // анализируемый текст
   unsigned int uOptions // опции обработки текста
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

pszHtml
[вх] Анализируемый текст, возможно с HTML-разметкой, в кодовой странице **Windows-1251**.

uOptions
[вх] Параметры предварительной очистки текста. Допустима комбинация следующих битовых флагов:

CA_OPT_DETECT_CODE_PAGE	Коррекция кодовой страницы текста.
CA_OPT_CLEAN_PAGE	Очистка HTML-страниц от навигационных элементов. На очистку страниц влияет пороговое значение, которое задается параметром *CA_P_DBL_LINK2TEXT_RATIO* (см. функцию **CAGetParameter**).

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Пример вызова

```
char* pszHtml =
    "<html><body>\n"
    "<p>Profile profiles profiling.\n"
    "</body></html>\n";
int nRes = CAProcessText( hCA, pszHtml, CA_OPT_CLEAN_PAGE |
   CA_OPT_DETECT_CODE_PAGE );
```

См. также

CAInitialize, **CAGetParameter**, **CAQueryInfo**

CAProcessTextEx

Функция категоризует текст, получение результатов – по одному заданному профилю. Их выводит функция **CAQueryInfo**, вызываемая с различными параметрами.

```
int CAProcessText(
   CA_HANDLE hCA, // дескриптор
   char* pszHtml, // анализируемый текст
   unsigned int uOptions, // опции обработки текста
   int* pCategory // указатель на ИД профиля
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

pszHtml
[вх] Анализируемый текст в кодировке, возможно с HTML-разметкой, в кодовой странице **Windows-1251**.

uOptions
[вх] Параметры предварительной очистки текста. Комбинация следующих битовых флагов влияет на точный анализ и синтез:

CA_OPT_DETECT_CODE_PAGE	Коррекция кодовой страницы текста.
CA_OPT_CLEAN_PAGE	Очистка HTML-страниц от навигационных элементов. На нее влияет пороговое значение, задаваемое параметром *CA_P_DBL_LINK2TEXT_RATIO* (см. функцию **CAGetParameter**).

pCategory
[вх] Указатель на идентификатор профиля. Если указатель задан, результаты категоризации формируются только для него. При выдаче он всегда стоит первым в списке вне зависимости от результатов категоризации (для других профилей они будут неопределенными). Если *pCategory* равен *0*, функция ведет себя, как и **CAProcessText**.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Пример вызова

```
char* pszHtml = "<html><body>\n <p>Profile profiles profiling.\n
   </body></html>\n";
int nTheOnlyCategory = 12345;
int nRes = CAProcessText( hCA, pszHtml, CA_OPT_CLEAN_PAGE |
   CA_OPT_DETECT_CODE_PAGE, &nTheOnlyCategory );
```

См. также

CAInitialize, **CAGetParameter**, **CAProcessText**, **CAQueryInfo**

CAGetAnnotation

При помощи функции составляется аннотация к тексту. При *pProcTermId = 0*, текст аннотируется по лексическому профилю, чей идентификатор определяется значением *nParam*. При *pProcTermId ≠ 0* аннотирование выполняется лишь по списку терминов, определяемым параметром *pProcTermId*, причем *nParam* задает их количество.

```
int CAGetAnnotation (
   CA_HANDLE hCA, // дескриптор
   int nParam // номер лексического профиля/количество терминов
   char** ppszAnnotation // указатель на текст аннотации
   int* pProcTermId // указатель на список терминов
   float* pProcTermWeight // указатель на массив весов терминов
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

nParam
[вх] идентификатор лексического профиля, по которому составляется аннотация.

ppszAnnotation
[вх] Указатель на результат аннотирования текста (библиотека выделяет этот буфер и управляет им). Указатель действителен до следующего вызова функции.

pProcTermId
[вх] Указатель на массив терминов для построения аннотации. Число элементов массива задается параметром *nParam*.

pProcTermWeight
[вх] Указатель на массив весов терминов для построения аннотации. Если он равен нулю, все термины считаются равноценными.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **`GetLastError`** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Пример вызова

```
char* pszAnnotation;
int nCategoryId = 0;
int Res1 = CAGetAnnotation( hCA, nCategoryId, &pszAnnotation, 0, 0);
vector<int> vProcTerm(1, 0);
int Res2 = CAGetAnnotation( hCA, vProcTerms.size(), &pszAnnotation,
   &vProcTerms[0], 0);
vector<float> vProcTermWeights(1, 1.5);
int Res3 = CAGetAnnotation( hCA, vProcTerms.size(), &pszAnnotation,
   &vProcTerms[0], &vProcTermWeights[0]);
*/
CAAPI(int) CAGetAnnotation( CA_HANDLE hCA, int nParam, char**
   ppszAnnotation, int* pProcTermId, float* pProcTermWeight );
```

CAQueryInfo

Функция **CAQueryInfo** позволяет получать информацию о результатах категоризации функциями **CAProcessText**, **CAProcessTextEx**, а также сообщения о нарушениях формата профиля после вызова **CALoadProfile**.

```
int CAQueryInfo(
   CA_HANDLE hCA, // дескриптор
   unsigned int uInfoType, // тип требуемой информации
   int nParam1, // вспомогательный параметр 1
   int nParam2, // вспомогательный параметр 2
   int nParam3, // вспомогательный параметр 3
   void* pResult // указатель на результат
);
```

Параметры

hCA
[вх] Дескриптор процедуры категоризации.

uInfoType
[вх] Тип запрашиваемой информации. Может принимать одно из следующих значений:

CA_Q_DOC_CLEAN_TEXT	Текст после очистки от навигационных элементов и HTML-разметки.
CA_Q_DOC_CLEAN_HTML	Текст с сохранением HTML-разметки после удаления навигационных элементов.
CA_Q_DOC_CATEGORY_COUNT	Число профилей, соотнесенных с текстом.
CA_Q_DOC_TRIED_CATEGORY_COUNT	Число профилей, имеющих общие термины с текстом.
CA_Q_PRF_LOADED_CATEGORY_COUNT	Общее число загруженных профилей.
CA_Q_PRF_CATEGORY_IDSTR	Строковый идентификатор рубрики.
CA_Q_PRF_CATEGORY_ID	Идентификатор профиля.
CA_Q_PRF_CATEGORY_STR	Название профиля.
CA_Q_PRF_CATEGORY_THR	Пороговое значение.
CA_Q_DOC_CATEGORY_SUM	Степень соответствия текста профилю.
CA_Q_DOC_TERM_COUNT	Число терминов профиля, найденных в тексте.
CA_Q_PRF_ALL_TERM_COUNT	Общее число терминов профиля.
CA_Q_PRF_TERM_STR	Строка термина.
CA_Q_PRF_TERM_WEIGHT	Весовой коэффициент термина.
CA_Q_DOC_TERM_FREQ	Частота термина в тексте.
CA_Q_DOC_TERM_SCORE	Вклад термина в степень соответствия текста профилю.
CA_Q_PRF_TERM_ID	Порядковый номер термина в профиле.
CA_Q_DOC_TERM_HIT_COUNT	Число элементов термина, найденных в тексте.
CA_Q_DOC_TERM_HIT_OFFSET	Смещение элемента термина.
CA_Q_DOC_TERM_HIT_LENGTH	Длина элемента термина.
CA_Q_DOC_CATEGORY_HIT_COUNT	Число элементов профиля, найденных в тексте.
CA_Q_DOC_CATEGORY_HIT_OFFSET	Смещение элемента профиля.
CA_Q_DOC_CATEGORY_HIT_LENGTH	Длина элемента профиля.
CA_Q_ERR_LOG_COUNT	Число нарушений формата профиля.
CA_Q_ERR_LOG_STR	Сообщение о нарушении формата профиля.

Значения и семантика остальных параметров зависят от типа запрашиваемой информации.

nParam1
[вх] Вспомогательный параметр.

nParam2
[вх] Вспомогательный параметр.

nParam3
[вх] Вспомогательный параметр.

pResult
[вых] Указатель на переменную, которой будет присвоен результат запроса. В зависимости от запроса переменная должна иметь тип **char***, **int** или **double**. В первом случае строка, на которую указывает переменная, связана с дескриптором и будет удалена при вызове **CAUninitialize**, а ее содержимое останется гарантировано неизменным до одного из следующих вызовов **CAProcessText**, **CAProcessTextEx** или **CALoadProfile**.

В случае неудачного завершения значение переменной, на которую указывает параметр *pResult*, будет неопределенным.

Возвращаемое значение

Список возвращаемых значений приведен в Таблице 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Unix**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечание. Функция работает со строками в кодовой странице **Windows-1251**.

Пример вызова

```
char* pszResult;
int nRes = CAQueryInfo( hCA, CA_Q_DOC_CLEAN_HTML, 0, 0, 0, &pszResult
   );
```

См. также

CAInitialize, **CALoadProfile**, **CAProcessText**, **CAGetAnnotation**, **CAUninitialize**

Работа с CAQueryInfo

В следующих разделах приведено описание возможных запросов к дескриптору процедуры категоризации посредством обращения к функции **`CAQueryInfo`**:

Текст после очистки от навигационных элементов и HTML-разметки

Текст с сохранением HTML-разметки после удаления навигационных элементов

Число профилей, соотнесенных с текстом

Число профилей, имеющих общие термины с текстом

Общее число загруженных профилей

Идентификатор профиля

Название профиля

Пороговое значение

Степень соответствия текста профилю

Число терминов профиля, найденных в тексте

Общее число терминов профиля

Строка термина

Весовой коэффициент термина

Частота термина в тексте

Вклад термина в степень соответствия текста профилю

Порядковый номер термина в профиле

Число элементов термина, найденных в тексте

Смещение элемента термина

Длина элемента термина

Число элементов профиля, найденных в тексте

Смещение элемента профиля

Длина элемента профиля

Число нарушений формата профиля

Сообщение о нарушении формата профиля

Текст после очистки от навигационных элементов и HTML-разметки

Текст после очистки от навигационных элементов и HTML-разметки. Используется для отладки параметров очистки текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CLEAN_TEXT, // тип требуемой информации
   0,
   0,
   0,
   &pszResult // указатель на переменную типа char*
);
```

Параметры

pszResult

[вых] Переменной *pszResult* будет присвоен указатель на буфер с текстом. Библиотека отвечает за выделение и освобождение памяти под буфер.

Пример вызова

```
char* pszResult;
CAQueryInfo( hCA, CA_Q_DOC_CLEAN_TEXT, 0, 0, 0, &pszResult );
```

Текст с сохранением HTML-разметки после удаления навигационных элементов

Текст с сохранением HTML-разметки после удаления навигационных элементов. Текст удаляемых навигационных элементов заменяется символами ‘x’ и ‘X’ для, соответственно, больших и малых букв. Используется для отладки параметров очистки текста и визуализации результатов очистки.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CLEAN_HTML, // тип требуемой информации
   0,
   0,
   0,
   &pszResult // указатель на переменную типа char*
);
```

Параметры

pszResult

[вых] Переменной *pszResult* будет присвоен указатель на буфер с текстом. Библиотека отвечает за выделение и освобождение памяти под буфер.

Пример вызова

```
char* pszResult;
CAQueryInfo( hCA, CA_Q_DOC_CLEAN_HTML, 0, 0, 0, &pszResult );
```

Число профилей, соотнесенных с текстом

Число профилей, соотнесенных с текстом.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CATEGORY_COUNT, // тип требуемой информации
   0,
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено число профилей, к которым был отнесен текст. Профили нумеруются от *1* до *nResult*.

Пример вызова

```
int nAssigned;
CAQueryInfo( hCA, CA_Q_DOC_CATEGORY_COUNT, 0, 0, 0, &nAssigned);
```

Замечания. Выполняется следующее соотношение между числом профилей, соотнесенных с текстом (*CA_Q_DOC_CATEGORY_COUNT*, обозначим как *nAssigned*), числом профилей, имеющих общие термины с текстом (*CA_Q_DOC_TRIED_CATEGORY_COUNT*, обозначим *nTried*), и общим числом загруженных профилей (*CA_Q_PRF_LOADED_CATEGORY_COUNT*, обозначим *nLoaded*):

0 <= nAssigned <= nTried <= nLoaded.

Профили в списке упорядочены по убыванию следующей величины, вычисляемой для каждого профиля:

0, если *score = threshold*;
score - threshold, если *score > threshold*;
max(score) - threshold, *score - threshold*, если *score < threshold*;
threshold - min(score), где

threshold – пороговое значение профиля (*CA_Q_PRF_CATEGORY_THR*), *score* – степень соответствия текста ему (*CA_Q_DOC_CATEGORY_SUM*), *min(score)* – теоретический минимум величины *score (<= 0)*, *max(score)* – теоретический максимум величины *score (>=min(score))*.
Если при расчете значение *threshold* находится вне интервала [*min(score)*, *max(score)*], оно приводится к ближайшей границе.
Данное выражение изменяется от *-1* до *1*. Его абсолютное значение – вероятность соответствия теме профиля, если величина положительна, и несоответствия, если отрицательная (в предположении равномерного распределения *score* среди положительных и отрицательных примеров).

Число профилей, имеющих общие термины с текстом

Число профилей, имеющих общие термины с текстом.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TRIED_CATEGORY_COUNT, // тип требуемой информации
   0,
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено число профилей, из которых в тексте нашелся хотя бы один термин, вне зависимости от результата отнесения текста к профилю. Профили нумеруются от *1* до *nResult*.

Пример вызова

```
int nTried;
CAQueryInfo( hCA, CA_Q_DOC_TRIED_CATEGORY_COUNT, 0, 0, 0, &nTried );
```

Замечания. Выполняется следующее соотношение между числом профилей, соотнесенных с текстом (*CA_Q_DOC_CATEGORY_COUNT*, обозначим как *nAssigned*), числом профилей, имеющих общие термины с текстом (*CA_Q_DOC_TRIED_CATEGORY_COUNT*, обозначим *nTried*), и общим числом загруженных профилей (*CA_Q_PRF_LOADED*_CATEGORY*_COUNT*, обозначим *nLoaded*):

0 <= nAssigned <= nTried <= nLoaded.

Профили в списке упорядочены по убыванию следующей величины, вычисляемой для каждого профиля:

0, если *score = threshold*;
score - threshold, если *score > threshold*;
max(score) – threshold
score - threshold, если *score < threshold*;
threshold - min(score), где

threshold – пороговое значение профиля (*CA_Q_PRF_CATEGORY_THR*), *score* – степень соответствия текста ему (*CA_Q_DOC_CATEGORY_SUM*), *min(score)* – теоретический минимум величины *score (<= 0)*, *max(score)* – теоретический максимум величины *score (>=min(score))*.
Если при расчете значение *threshold* находится вне интервала [*min(score)*, *max(score)*], оно приводится к ближайшей границе.
Данное выражение изменяется от *-1* до *1*. Его абсолютное значение – вероятность соответствия теме профиля, если величина положительна, и несоответствия, если отрицательная (в предположении равномерного распределения *score* среди положительных и отрицательных примеров).

Общее число загруженных профилей

Общее число загруженных профилей.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_LOADED_CATEGORY_COUNT, // тип требуемой информации
   0,
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено общее число загруженных профилей. Профили нумеруются от *1* до *nResult*.

Пример вызова

```
int nLoaded;
CAQueryInfo( hCA, CA_Q_PRF_LOADED_CATEGORY_COUNT, 0, 0, 0, &nLoaded
   );
```

Замечание. Выполняется следующее соотношение между числом профилей, соотнесенных с текстом (*CA_Q_DOC_CATEGORY_COUNT*, обозначим как *nAssigned*), числом профилей, имеющих общие термины с текстом (*CA_Q_DOC_TRIED_CATEGORY_COUNT*, обозначим *nTried*), и общим числом загруженных профилей (*CA_Q_PRF_LOADED_CATEGORY_COUNT*, обозначим *nLoaded*):

$$0 <= nAssigned <= nTried <= nLoaded$$

См. также

CALoadProfile

Идентификатор профиля

Служит для сопряжения результатов работы библиотеки с решаемой прикладной задачей.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_CATEGORY_ID, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоен идентификатор профиля.

Пример вызова

```
int nId;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  CAQueryInfo( hCA, CA_Q_PRF_CATEGORY_ID, nParam1, 0, 0, &nId );
```

Замечание. Идентификатор профиля задается либо при загрузке (см. описание **CALoadProfile**), либо в теле профиля (описание формата см. в Приложении 1).

Название профиля

Название профиля. Задается в теле профиля (см. Приложение 1, **CAQueryInfo**).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_CATEGORY_STR, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &pszResult // указатель на переменную типа char*
);
```

Параметры

nParam1

[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

pszResult

[вых] Переменной *pszResult* будет присвоен указатель на буфер с названием профиля. Библиотека отвечает за выделение и освобождение памяти под буфер.

Пример вызова

```
char* pszName;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  CAQueryInfo( hCA, CA_Q_PRF_CATEGORY_STR, nParam1, 0, 0, &pszName );
```

Пороговое значение

Служит для отнесения текста к профилю, задается в теле профиля (см. Приложение 1).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_CATEGORY_THR, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &dblResult // указатель на переменную типа double
);
```

Параметры

nParam1
[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

dblResult

[вых] Переменной присваивается пороговое значение для отнесения текста к профилю.

Пример вызова

```
double dblThr;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
 CAQueryInfo( hCA, CA_Q_PRF_CATEGORY_THR, nParam1, 0, 0, &dblThr );
```

Степень соответствия текста профилю

Вычисляется при разборе текста, используется для сравнения с пороговым значением.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CATEGORY_SUM, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &dblResult // указатель на переменную типа double
);
```

Параметры

nParam1
[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

dblResult

[вых] Переменной присваивается пороговое значение для отнесения текста к профилю.

Пример вызова

```
double dblScore;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
 CAQueryInfo( hCA, CA_Q_DOC_CATEGORY_SUM, nParam1, 0, 0, &dblScore );
```

См. также

CAQueryInfo

Число терминов профиля, найденных в тексте

Число терминов профиля, найденных в тексте. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_COUNT, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено число терминов профиля, найденных в тексте.

Пример вызова

```
int nTermCount;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  CAQueryInfo( hCA, CA_Q_DOC_TERM_COUNT, nParam1, 0, 0, &nTermCount
   );
```

Замечание. Между числом терминов профиля, найденных в тексте (*CA_Q_DOC_TERM_COUNT*, обозначим *TermCount*), и общим числом терминов профиля (*CA_Q_PRF_ALL_TERM_COUNT*, обозначим *nAllTermCount*) существует отношение:

$$0 <= nTermCount <= nAllTermCount$$

Термины при перечислении упорядочены по убыванию вклада в степень соответствия текста профилю (*CA_Q_DOC_TERM_SCORE*).

Общее число терминов профиля

Общее число терминов профиля. Определяется при загрузке профиля (см. Приложение 1).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_ALL_TERM_COUNT, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля (начинается с 1). Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено общее число терминов профиля.

Пример вызова

```
int nAllTermCount;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  CAQueryInfo( hCA, CA_Q_PRF_ALL_TERM_COUNT, nParam1, 0, 0,
   &nAllTermCount );
```

Замечание. Между числом терминов профиля, найденных в тексте (*CA_Q_DOC_TERM_COUNT*, обозначим *TermCount*), и общим числом терминов профиля (*CA_Q_PRF_ALL_TERM_COUNT*, обозначим *nAllTermCount*) существует отношение:

$$0 <= nTermCount <= nAllTermCount$$

Термины при перечислении упорядочены по убыванию вклада в степень соответствия текста профилю (*CA_Q_DOC_TERM_SCORE*).

Строка термина

Строка термина. Определяется при загрузке профиля (см. Приложение 1).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_TERM_STR, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &pszResult // указатель на переменную типа char*
);
```

Параметры

nParam1

[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2

[вх] Индекс термина. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

pszResult

[вых] Переменной *pszResult* будет присвоен указатель на буфер со строкой термина. Библиотека отвечает за выделение и освобождение памяти под буфер.

Пример вызова

```
char* pszTermStr;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_PRF_TERM_COUNT, nParam1, nParam2, 0,
   &pszTermStr );
```

Весовой коэффициент термина

Весовой коэффициент термина. Определяется при загрузке профиля (см. Приложение 1).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_TERM_WEIGHT, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &dblResult // указатель на переменную типа double
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс термина. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

dblResult

[вых] Переменной *dblResult* будет присвоен весовой коэффициент термина.

Пример вызова

```
double dblTermWeight;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_PRF_TERM_COUNT, nParam1, nParam2, 0,
   &dblTermWeight );
```

Частота термина в тексте

Частота термина в тексте. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_FREQ, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс термина. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоена частота термина в тексте.

Пример вызова

```
int nTermFreq;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_DOC_TERM_COUNT, nParam1, nParam2, 0,
   &nTermFreq);
```

Замечание. В случае если термин является словом или словосочетанием, частота термина совпадает с количеством его вхождений в текст.

Если термин является логическим выражением, частота считается по следующим правилам:

1. Freq (A & B) = min Freq(A), Freq(B);

2. Freq (A | B) = max Freq(A), Freq(B);

3. Freq (A ~ B) = min Freq(A), Freq(B).

Вклад термина в степень соответствия текста профилю

Вклад термина в степень соответствия текста профилю. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_SCORE, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &dblResult // указатель на переменную типа double
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс термина. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

dblResult

[вых] Переменной *dblResult* будет присвоен вклад термина в степень соответствия текста профилю.

Пример вызова

```
double dblTermComp;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_TDOC_ERM_SCORE, nParam1, nParam2, 0,
   &dblTermComp );
```

Порядковый номер термина в профиле

Порядковый номер термина в профиле. Определяется при загрузке профиля (см. Приложение 1).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_PRF_TERM_ID, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2

[вх] Индекс термина. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоен порядковый номер термина в профиле.

Пример вызова

```
int nTermId;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_PRF_TERM_ID, nParam1, nParam2, 0, &nTermId
   );
```

Число элементов термина, найденных в тексте

Число элементов термина – слов и словосочетаний, составляющих термин, найденных в тексте. Вычисляется в процессе разбора текста. Используется для визуализации результатов категоризации (подсветки релевантных фрагментов текста).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_HIT_COUNT, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2

[вх] Индекс термина. Начинается с 1. Число терминов можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено число элементов термина.

Пример вызова

```
int nTermHitCount;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_DOC_TERM_HIT_COUNT, nParam1, nParam2, 0,
   &nTermHitCount );
```

Замечание. Элементы термина представлены как непересекающиеся фрагменты исходной HTML-разметки, покрывающие слова и словосочетания, из которых состоит термин, упорядоченные по возрастанию смещения в тексте.

Смещение элемента термина

Смещение элемента термина – слова, словосочетания. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_HIT_OFFSET, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   nParam3, // индекс элемента термина
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2

[вх] Индекс термина. Начинается с 1. Число терминов можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

nParam3

[вх] Индекс элемента термина. Начинается с 1. Число терминов можно получить путем запроса значения *CA_Q_DOC_TERM_HIT_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено смещение элемента термина.

Пример вызова

```
int nTermHitOffset;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    for (int nParam3 = 1; nParam3 <= nTermHitCount; nParam3++)
      CAQueryInfo( hCA, CA_Q_DOC_TERM_HIT_OFFSET, nParam1, nParam2,
   nParam3, &nTermHitOffset );
```

Длина элемента термина

Длина элемента термина – слова, словосочетания. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_TERM_HIT_LENGTH, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс термина
   nParam3, // индекс элемента термина
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс термина. Начинается с 1. Число терминов можно получить путем запроса значений *CA_Q_DOC_TERM_COUNT* и *CA_Q_PRF_ALL_TERM_COUNT*.

nParam3
[вх] Индекс элемента термина. Начинается с 1. Число терминов можно получить путем запроса значения *CA_Q_DOC_TERM_HIT_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоена длина элемента термина.

Пример вызова

```
int nTermHitLength;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nTermCount; nParam2++)
    for (int nParam3 = 1; nParam3 <= nTermHitCount; nParam3++)
      CAQueryInfo( hCA, CA_Q_DOC_TERM_HIT_LENGTH, nParam1, nParam2,
   nParam3, &nTermHitLength );
```

Число элементов профиля, найденных в тексте

Число элементов всех терминов профиля (слов и словосочетаний), составляющих все термины профиля, найденных в тексте. Вычисляется в процессе разбора текста. Используется для визуализации результатов категоризации (подсветки релевантных фрагментов текста).

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CATEGORY_HIT_COUNT, // тип требуемой информации
   nParam1, // индекс профиля
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено число элементов всех терминов профиля.

Пример вызова

```
int nCategoryHitCount;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  CAQueryInfo( hCA, CA_Q_DOC_TERM_HIT_COUNT, nParam1, nParam2, 0,
   &nCategoryHitCount );
```

Замечание. Элементы терминов профиля представлены как непересекающиеся фрагменты исходной HTML-разметки, покрывающие слова и словосочетания, из которых состоит термин, упорядоченные по возрастанию смещения в тексте.

Данный запрос позволяет сделать «подсветку» одновременно всех терминов профиля, иллюстрируя принятие решения об отнесении текста к профилю.

Смещение элемента профиля

Смещение элемента профиля – слова, словосочетания. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CATEGORY_HIT_OFFSET, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс элемента профиля
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс элемента термина. Начинается с 1. Число терминов можно получить путем запроса значения *CA_Q_DOC_CATEGORY_HIT_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоено смещение элемента профиля.

Пример вызова

```
int nCategoryHitOffset;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nCategoryHitCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_DOC_CATEGORY_HIT_OFFSET, nParam1, nParam2,
   0, &nCategoryHitOffset );
```

Длина элемента профиля

Длина элемента профиля – слова, словосочетания. Вычисляется в процессе разбора текста.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_DOC_CATEGORY_HIT_LENGTH, // тип требуемой информации
   nParam1, // индекс профиля
   nParam2, // индекс элемента профиля
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nParam1
[вх] Индекс профиля. Начинается с 1. Число профилей можно получить путем запроса значений *CA_Q_DOC_CATEGORY_COUNT*, *CA_Q_DOC_TRIED_CATEGORY_COUNT*, *CA_Q_PRF_LOADED_CATEGORY_COUNT*.

nParam2
[вх] Индекс элемента термина. Начинается с 1. Число терминов можно получить путем запроса значений *CA_Q_DOC_CATEGORY_HIT_COUNT*.

nResult

[вых] Переменной *nResult* будет присвоена длина элемента профиля.

Пример вызова

```
int nCategoryHitlength;
for (int nParam1 = 1; nParam1 <= nTried; nParam1++)
  for (int nParam2 = 1; nParam2 <= nCategoryHitCount; nParam2++)
    CAQueryInfo( hCA, CA_Q_DOC_CATEGORY_HIT_LENGTH, nParam1, nParam2,
   0, &nCategoryHitlength );
```

Число нарушений формата профиля

Число нарушений формата профиля. Подсчитывается в процессе загрузки профиля.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_ERR_LOG_COUNT, // тип требуемой информации
   0,
   0,
   0,
   &nResult // указатель на переменную типа int
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено число нарушений формата профиля.

Пример вызова

```
int nErrCnt;
CAQueryInfo( hCA, CA_Q_ERR_LOG_COUNT, 0, 0, 0, &nErrCnt );
```

Замечание. После загрузки очередного профиля все предыдущие сообщения об ошибках стираются.

Сообщение о нарушении формата профиля

Сообщение о нарушении формата профиля. Создается в процессе загрузки профиля.

```
int CAQueryInfo(
   hCA, // дескриптор
   CA_Q_ERR_LOG_STR, // тип требуемой информации
   nParam1, // индекс сообщения
   0,
   0,
   &pszResult // указатель на переменную типа char*
);
```

Параметры

nParam1

[вх] Индекс сообщения. Начинается с 1. Число нарушений формата профиля можно получить путем запроса *CA_Q_ERR_LOG_COUNT*.

pszResult

[вых] Переменной *pszResult* будет присвоен указатель на буфер с сообщением о нарушении формата профиля. Библиотека отвечает за выделение и освобождение памяти под буфер.

Пример вызова

```
char* pszErrStr;
for (int nParam1 = 1; nParam1 <= nErrCnt; nParam1++)
  CAQueryInfo( hCA, CA_Q_ERR_LOG_STR, nParam1, 0,0, &pszErrStr);
```

Пример использования

Поставляемый с библиотекой пример (**casample.cpp**) выполняет следующие действия:

1. Принимает на вход параметры:

   – Путь к каталогу с профилями (см. Приложение 1);

   – Путь к каталогу с текстами;

   – Путь к каталогу с результатами категоризации.

2. Загружает профили (в качестве профилей воспринимаются файлы с расширением *.txt);

3. Сканирует каталог с текстами (*.htm) и для каждого файла вызывает `CAProcessText`;

4. Сохраняет результаты категоризации в следующем виде:

   – Текст с сохранением HTML-разметки после очистки навигационных элементов: **<ИСХ_ФАЙЛ>.clean.htm**;

   – Текст после удаления HTML-разметки и очистки навигационных элементов: **<ИСХ_ФАЙЛ>.clean.txt**;

   – Текст с подсветкой всех элементов всех профилей **<ИСХ_ФАЙЛ>.highlight.htm**;

   – Отчет о результатах сопоставления с каждой категорией и терминами каждой из категорий, найденными в тексте: **<ИСХ_ФАЙЛ>.report.htm**.

Приложение 1. Описание формата профиля

Номер строки	Данные
1	Идентификатор профиля – если есть, должен быть конвертируем в целое число (int)
2	Название профиля – может быть пустым (string)
3	Способ интерпретации профиля (string)
4	Пороговое значение – число с плавающей точкой (double)
5	// reserved
6	// reserved
7	// reserved
8	// reserved
9	// reserved
10	
11	Количество терминов в профиле – целое число (int)
12	Описание термина 1
...	
11+N	Описание термина N
12+N	
13+N	Количество используемых парадигм
14+N	Парадигма 1
...	
13+N+M	Парадигма M

Способ интерпретации профиля ::=

```
<Способ взвешивания>"|"<Нормировка>
```

Способ взвешивания:=

```
"binary"   (термин есть/нет)
"frequency"   (учет частоты термина)
"bayes"..(нелинейное байесовское взвешивание с пуассоновской
 нормировкой)[1]
```

Нормировка ::=

```
"document_length_normalization"   (число слов документа – не
 применима при способе взвешивания binary)
"profile_based_normalization"   (евклидова норма)
"poisson"   (пуассоновская норма)[1]
```

Описание термина ::=

```
<весовой_коэффициент(double)> <tab> <тип_термина> <tab>
 <строка_термина>
```

Тип термина ::=

```
"*" (обычный – может встретиться, а может – нет)
"+" (обязательный – должен встретиться)
"-" (исключающий – не должен встретиться)
"?" (условный – дает вклад в степень соответствия, но для отнесения
 документа к рубрике он должен содержать хотя бы один обычный или
 обязательный термин)
```

Строка термина ::=

```
"!"?<слово>
"!"?<словосочетание>
<словосочетание> "/"<размер окна> (в строке термина запрещено
 использование других операторов)
"(" <строка_термина> ")"
<строка_термина> <оператор> <строка_термина>
```

Оператор ::=

```
"&" (лог «И»)
"|" (лог «ИЛИ»)
"~" (лог «И НЕ»)
```

Экранирующий символ "\" – после него любой служебный символ считывается как текст. Список служебных символов в строке термина: "&", "|", "~", "(", ")", "/", "\".

Парадигма ::=

```
<слово> ("=" <слово>)*
```

[1] Плешко В.В., Поляков П.Ю., Ермаков А.Е.; RCO на РОМИП 2009 // Труды РОМИП 2009 (Петрозаводск, 2009 г.); Санкт-Петербург: НУ ЦСИ, 2009, с. 122-134

```
<словосочетание> ("=" <словосочетание>)*
```

Пояснения:

1. Термином может быть слово, словосочетание или логическое выражение;

2. Для слов и словосочетаний в конце профиля могут быть приведены парадигмы (варианты написаний, которые должны быть распознаны в тексте);

3. Если парадигма не найдена, слово/словосочетание распознается в зависимости от того, включен ли морфоанализатор. Если да – распознаются все словоформы, нет – распознается в соответствии с записью в строке термина;

4. Если перед словом/словосочетанием стоит модификатор "!", то слово/словосочетание распознается в соответствии с записью в строке термина;

5. Если в конце словосочетания стоит конструкция /N, то между словами словосочетания допускаются другие слова с ограничением на длину последовательности слов текста (N), в которой должны встретиться все слова словосочетания;

6. Перечень парадигм задается для каждого профиля отдельно;

7. В парадигме не обязательно указывать все словоформы;

8. Термин вставляется всегда, то есть допускаются одинаковые термины;

9. Дубликаты парадигм запрещены. Первая словоформа в парадигмах служит ключом. Не допускается двух парадигм с одинаковыми ключами, все следующие парадигмы с тем же ключом указывают на первую;

10. Все не тексто-цифровые конструкции (не содержащие хотя бы одну букву или цифру) при обработке текста игнорируются;

11. Слова должны быть написаны без знаков препинания и кавычек, иначе не будут найдены в тексте (дефис не является знаком препинания).

Пример профиля:

```
12345
Прострой профиль
binary|profile_length
1.0
// reserved
// reserved
// reserved
// reserved
// reserved

4
0.6   *     мама мыла раму
0.5   +     папа & кошка сидела на заборе
0.4   +     вилка | ложка
0   -  большое окно

2
папа=папе=папу=папой
большое окно=большому окну
```

Примеры терминов:

```
папа & кошка сидела на заборе
вилка на тарелке | ложка в стакане
вилка на тарелке | !сосулька на стене
(чашка & кружка) | стакан стоит на столе
(чашка & кружка) стакан стоит на столе      --- ошибка
медведь ~ рыба
медведь ~рыба
медведь~рыба
```