УДК 004.89+004.912

Колешко В.М., Шиманович Е.А.

ОПРЕДЕЛЕНИЕ ЯЗЫКА ТЕКСТА В ИНТЕЛЛЕКТУАЛЬНОЙ ЛИНГВИСТИЧЕСКОЙ СИСТЕМЕ

Белорусский национальный технический университет
Минск, Беларусь

Разработка интеллектуальных лингвистических систем синхронизированных многоязычных систем является перспективной задачей XXI века. Это обусловлено, во-первых, тем, что не каждый человек обладает лингвистическими талантами, во-вторых, даже талантливый затрачивает много усилий и времени по изучению «литературно-разговорных» иностранных языков. В-третьих, XXI век требует от каждого цивилизованного человека успешно владеть не одним иностранным языком, не говоря уже о родном языке. Особенно необходимы такие лингвистические системы студенту, поэтому задачу, которую мы поставили перед собой – разработка интеллектуально-лингвистической системы синхронного многоязычного перевода, является сложной, но решаемой, особенно с участием профессорских и студенческих интеллектуальных усилий. Увеличивает данную уверенность то, что в последнее время сеть Интернет стала широко распространена во всем мире и содержит в себе гигантское количество информации на различных языках. Данная программа может быть использована для классификации этой информации, кроме того, она может быть использована как модуль в других пакетах программного обеспечения, например, при проверке орфографии и прочей обработке естественных языков.

Несколько ученых из Амстердамского университета заметили интересную вещь[1]: если взять текст на некотором языке, скажем английском, заархивировать его обычным популярным архиватором WinZip, затем взять этот же текст, только прибавить к нему какой-то фрагмент тоже на английском языке, и вновь запаковать, то размеры файлов будут практически одинаковыми. Но если добавить текст на другом языке, скажем, немецком, то размер конечного архива значительно возрастает. Было проведено несколько тестов для различных языков, причем были получены удивительно хорошие результаты: такая ситуация наблюдалась в 100% случаев для всех языков Евросоюза; минимальный размер добавляемого текста должен был быть около сотни символов.

Алгоритм архивации. Если вникнуть в суть работы алгоритма, то такая картина становится неудивительной. WinZip использует алгоритмs LZW (Lempel-Ziv-Welch) - усовершенствованный Уэлшем алгоритм LZ , имеющий разные варианты исполнения LZ77, LZ78. Это алгоритмы сжатия без потерь, опубликованные в статьях Абрама Лемпела (Abraham Lempel) и Якоба Зива (Jacob Ziv) в 1977 и 1978. [2] Эти два алгоритма являются наиболее известными вариантами в семействе LZ, которое также включает в себя LZW, LZSS, LZMA и другие алгоритмы. Основной принцип работы алгоритмов этого семейства в следующем: в соответствии с поступающими на вход словами-символами алгоритм заносит специальным образом построенные фразы в свой словарь, и затем, когда встречает во входном потоке эти фразы, то заменяет их на индекс в словаре. Почти все существующие алгоритмы этого класса отличаются в основном лишь способом ведения этого словаря.

По сути, данный алгоритм формирует словарь, который содержит набор характерных для языка сочетаний символов. Как показали эксперименты, для разных языков эти словари значительно отличаются, что и объясняет незначительный прирост размера архива при добавлении к тексту фрагмента на таком же языке, т.к. уже сформированный словарь отлично подходит для архивации добавленного фрагмента.

Рассмотрим более подробно сам алгоритм, точнее его часть, в которой формируется словарь. Давайте сначала определим нечто, называемое "текущим префиксом". Обозначим его как "prefix". Изначально текущий префикс ничего не содержит. Давайте также определим "текущую цепочку", которая образуется текущим префиксом и следующим символом в потоке символов. Обозначим текущую цепочку как "prefix+C", где C - некоторый символ.

Теперь посмотрим на первый символ в потоке символов. Назовем его P. Сделаем prefix+P текущей цепочкой. Теперь выполним поиск в таблице кодовых слов, чтобы определить входит ли в нее prefix+P. Конечно, сейчас это не произойдет, поскольку наша таблица пустая. В этом случае

мы добавляем текущую цепочку в таблицу, а префикс делаем пустым. Снова делаем prefix+P текущей цепочкой, где P – следующий символ потока. Через несколько шагов текущая цепочка найдется в кодовом словаре. Тогда значение текущей цепочки присваивается префиксу: prefix = prefix +P, где P – очередной символ. А текущая цепочка вновь формируется с уже новым префиксом и очередным символом потока. При архивировании словарь кодовых слов является динамичным, т.е. по мере наполнения словаря, новые слова вытесняют старые. Т.к. у нас стоит задача лишь сформировать словарь некоторого заданного размера, то по достижении этого размера мы прекращаем работу алгоритма. Стоит отметить, что для формирования словаря нужен некоторый текст минимальной длины, который позволил бы заполнить весь словарь полностью, а также то, что нет необходимости обучать систему на больших объемах текста. Необходим такой размер текста, который обеспечил бы лишь полное формирование словаря.

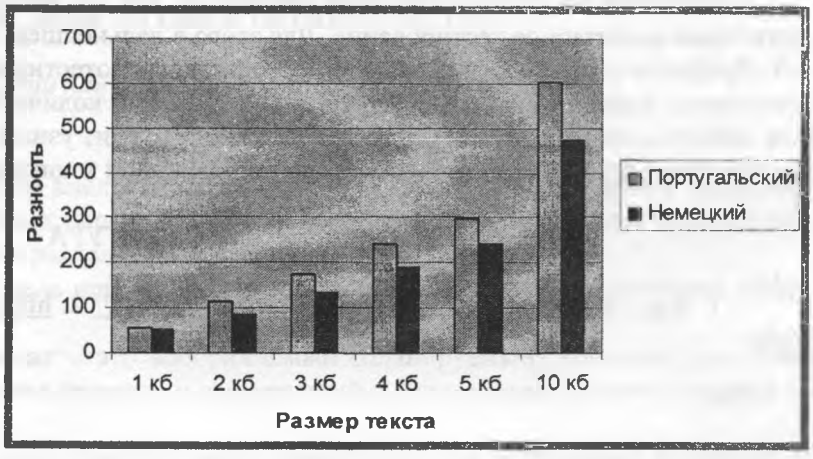

Рисунок 1. Разность между английским и немецким/португальским словарями.

Алгоритм работы системы. Работу с системой в данной работе можно разделить на два этапа: 1. Обучение; 2. Непосредственно распознавание.

Обучение подразумевает собой то, что мы формируем словари для всех языков, которым нам нужно обучить систему. Способ формирования словаря был приведен выше. Следует заметить, что системе нет необходимости обучаться на образцах очень больших размеров, т.к. в системе используется не статистические показатели наподобие частоты встречаемости отдельных символов, а характерные для языка буквосочетания.

Распознавание заданного текста происходит следующим образом: поочередно используя каждый словарь из уже сформированного на 1 этапе набора словарей, имитируется процесс кодирования заданного текста. Реального сжатия текста не происходит, т.к. это не является целью данного этапа разработки. Как уже упоминалось выше, алгоритм кодирования пытается заменять фрагменты текста ссылками на кодовые слова из словаря.

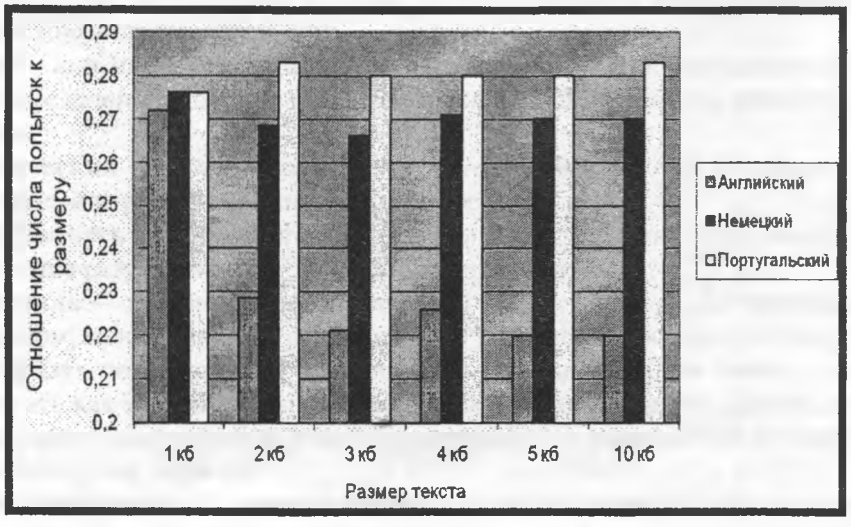

Рисунок 2. Отношение числа попыток к размеру текста.

Данная система подсчитывает неудачные попытки замены фрагментов текста на ссылки. На выходе мы получаем массив размерности, равный количеству словарей (т.е. языков распознавания), отсортировав который по возрастанию мы получим очередность того, насколько введенный текст «похож» на языки из нашей коллекции словарей. Т.е. наименьшее число неудачных попыток замены соответсвует наибольшей степени похожести.

Тестирование. Для тестирования разработанная программа была обучена трем языкам: английский, немецкий и португальский. Далее на каждом языке были созданы по 10 текстов размером от 1Кб. Источником текстов для английского и португальского послужил сайт www.bbc.com, для немецких – новостной сайт www.magazine-deutschland.de . Все тексты были распознаны правильно, т.е. программа показала 100% вероятность определения языка. Также на

примере текста на английском программа была протестирована для разных размеров текста (от 1кб до 10кб). На рисунке 1 приведена диаграмма разности неудачных попыток между английским словарем с немецким и португальским словарем.

Как можно видеть, этот показатель зависит практически линейно от размера текста. Отношение числа попыток ко всему размеру текста для всех 3-х словарей приведено на рисунке 2.

Можно видеть, что результаты становятся довольно надежными при размере текста около 2Кб и более.

Несмотря на обнадеживающие результаты для 3-х языков и 30 текстов все же стоит провести более масштабное тестирование. Для этого в дальнейшем нужно добавить поддержку пакетной обработки текстовых файлов, что позволило бы протестировать разработанную программу на значительно больших массивах файлов и для большего количества языков. Это поможет исследовать особенности работы алгоритма и возможно позволит узнать другие интересные закономерности, что будет приближать нас к решению поставленной многоэтапной задачи.

ЛИТЕРАТУРА

1. http://www.computerra.ru/news/2003/4/14/38987/ 2. http://en.wikipedia.org/wiki/Lempel-Ziv-Welch