УДК 004.522

# Речевой интерфейс для интеллектуализации ввода компьютерных программ

В.С. Бакаленко, О.И. Федяев
Донецкий национальный технический университет
fedyaev@r5.dgtu.donetsk.ua
valeriy.bakalenko@gmail.com

***Бакаленко В.С., Федяев О.И. Речевой интерфейс для интеллектуализации ввода компьютерных программ.*** *В статье описывается система речевого ввода текста программы на языке программирования. Речевой интерфейс представляется программистам более естественный способ набора исходного кода программы. Ядром интерфейса является акустико-лингвистическая модель автоматического распознавания речи.*
***Ключевые слова:*** *речь, Sphinx, распознавание, синтез.*

## *Введение*

В настоящее время набор текста программы на языке программирования выполняется вручную с помощью клавиатуры. Процедура ввода требует хороших навыков работы с клавиатурой, большого внимания и напряжения на зрение. Такой способ ввода для человека является трудоёмким и не совсем удобным [1-3]. Этот недостаток можно устранить путём качественного решения задачи автоматического распознавания речи [4].

Основными проблемами в решении этой задачи являются: большой объём словаря используемых слов; высокая скорость распознавания; качество распознавания. Кроме того, необходимо учитывать многообразие вариантов названий переменных и функций. Скорость распознавания должна быть приемлемой, чтобы программист мог диктовать код и видеть в реальном времени предварительные результаты распознавания [5]. От точности распознавания зависит результат компиляции исходного кода и время исправления ошибок.

3 Поэтому цель данной работы состоит в разработке речевого интерфейса, обеспечивающего взаимодействие человека с компьютером более естественным способом. Задача ввода текста программ голосом решается с помощью технологии Sphinx. Она обеспечивает возможность построения акустико-лингвистической модели для любого языка программирования и настройки на особенности голоса диктора.

## *Инструментальная среда CMU Sphinx4*

На сегодняшний день Sphinx [4], является самым популярным и работоспособным из открытых движков. Он разработан в университете Карнеги-Меллона с участием Массачусетского технологического института и Sun Microsystems. CMU Sphinx распространяется на условиях лицензии BSD, т. е. доступен для коммерческого и некоммерческого использования. Одним из достоинств Sphinx является поддержка множества языков и способность создавать свои собственные модели распознавателей речи.

Sphinx-4, как версия из семейства CMU Sphinx [6], состоит из двух компонентов: «тренера» (trainer) и декодера. Тренер создаёт акустическую модель, адаптированную под конкретные потребности, а декодер выполняет собственно распознавание. Архитектура Sphinx-4 на верхнем уровне относительно проста. Как показано на рисунке 1, она включает FrontEnd, клиентскую часть (приложение), декодер и базу знаний.

Блок FrontEnd отвечает за сбор, аннотирование и обработку входных данных. Кроме того, он извлекает объекты из входных данных для чтения с помощью декодера. Аннотации, предусмотренные во FrontEnd, включают в себя начальный и конечный сегменты данных. Операции, выполняемые на данном этапе, реализуют шумоподавление, автоматическую регулировку усиления, анализ Фурье, спектральную фильтрацию Мэла и др.

База знаний содержит информацию необходимую для декодера. Эта информация включает в себя акустическую модель и модель языка. В свою очередь декодер может послать команду базе знаний, требующую от базы знаний динамически изменять себя на основе результатов поиска. Эти модификации могут заключаться в переключении акустических моделей и/или языка модели, а также обновлении некоторых параметров, например, дисперсии преобразования для акустических моделей.

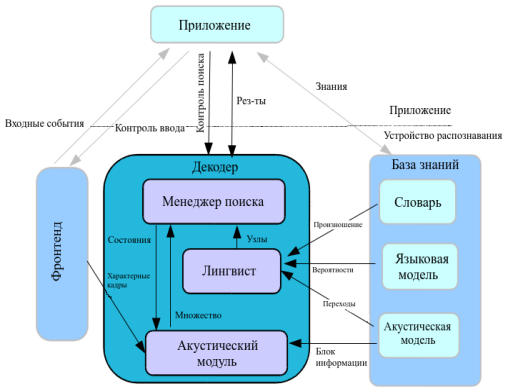



Рисунок 1 – Структура инструментария CMU Sphinx4

Декодер выполняет основную часть работы. Он считывает данные с FrontEnd, сопоставляет их с данными из базы знаний и откликом приложения, а также выполняет поиск в пространстве наиболее вероятных последовательностей слов, которые входят в число претендентов на выбор. Термин «пространство поиска» означает описание наиболее вероятных последовательностей слов, которые динамически обновляются с помощью декодера в процессе декодирования.

В отличие от множества других архитектур распознавателей речи Sphinx-4 позволяет приложению контролировать множество функций речевого движка. Во время декодирования приложение может получать данные от декодера. Эти данные позволяют приложению отслеживать, как происходит процесс декодирования, а также влиять на процесс декодирования до его завершения. Кроме того, приложение может обновлять базу знаний в любое время.

Но главным достоинством Sphinx является возможность описания проектируемого распознавателя на уровне формальных моделей, что и послужило основанием для выбора Sphinx-4 в качестве инструментария.

Лингвистическая модель является основой построения приложения распознающего и синтезирующего речь.

Лингвистическая модель состоит из словаря, языковой модели, списка фонем и прочих звуков, обучающего множества. Она служит входными данными для построения акустической модели. В словаре содержится список слов и транскрипции к ним. Транскрипции должны состоять исключительно из фонем, которые присутствуют в списке фонем. Помимо слов есть и другие звуки, не несущие в себе смысловой нагрузки: звуки дыхания, различный шум. Языковая модель – совокупность вероятностей появления слов в речи. В соответствии со всеми вышеперечисленными характеристиками речи записывается аудиобаза и она оформляется в виде обучающих примеров с правильными ответами. Каждая аудиозапись должна иметь своё текстовое представление. Чем больше материала в аудиобазе, тем лучше качество распознавания. Элементы информационной структуры лингвистической модели показаны на рисунке 2.

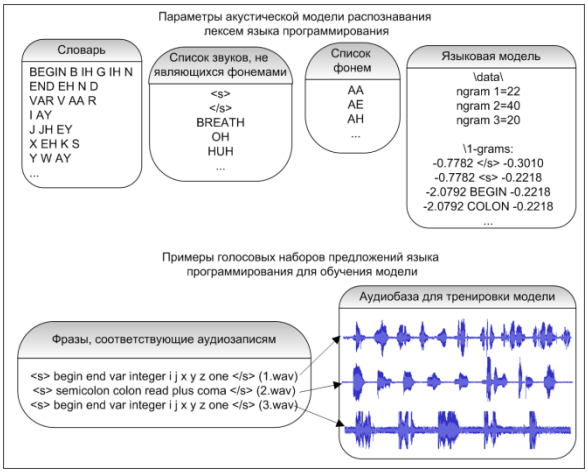



Рисунок 2 – Информационная структура лингвистической модели

### *Общая схема распознавания и синтеза речи*

Взаимодействие человека и компьютера осуществляется с помощью клавиатуры, мыши, монитора, микрофона и колонок. С помощью микрофона и подсистемы отвечающей за распознавание речи пользователь может вводить голосом тексты программ на языке программирования Pascal. Также имеется возможность производить ввод текста программы с помощью клавиатуры. Речевой синтез текстов программ Pascal осуществляется с помощью колонок или наушников и при помощи подсистемы синтеза речи. Кроме того, можно настроить скорость и громкость чтения текстов программ.

Человек отвечает за ввод текстов программ с помощью клавиатуры либо микрофона. Компьютер занимается распознаванием речи, ее синтезом и выводом результатов на формы приложения, отображаемые на мониторе. Во время работы, приложение выводит различную служебную информацию о ходе распознавания речи.

Благодаря разработанному программному продукту пользователь может вводить тексты программ не только с помощью клавиатуры, но и с помощью речи. Чтобы не напрягать зрение и проверять программу, пользователь может просто прослушать написанную им программу.

Функциональная схема разработанного приложения изображена на рисунке 3.


Рисунок 3 – Общая схема речевого интерфейса

### *Разработка речевого интерфейса*

Разрабатываемый интерфейс внешне выглядит как приложение-блокнот, сохраняющий возможность ввода текста с помощью клавиатуры, а так же позволяющий вводить тексты программ с помощью речи, а затем сохранять их, озвучивать и отправлять на компиляцию [7].

Интерфейс приложения, изображенный на рисунке 4, состоит из строки заголовка, строки меню, рабочей области, кнопок для активирования режимов синтеза и распознавания речи, а также небольшой области с выводом сообщений о состоянии инструмента распознавания.

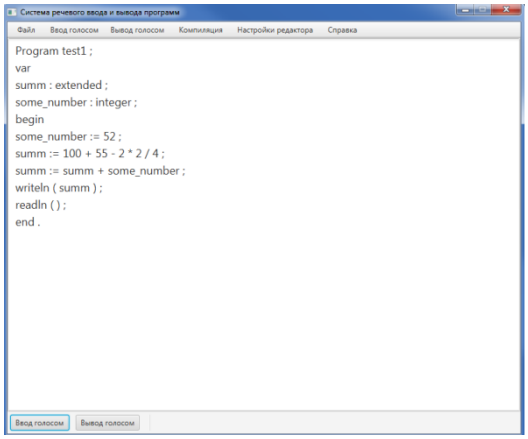
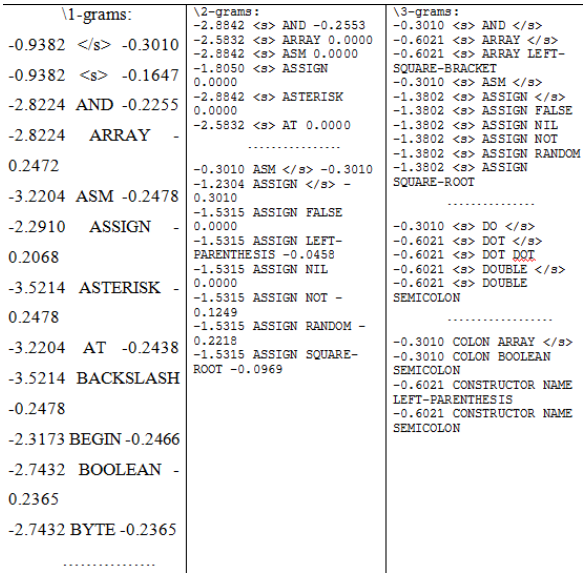



Рисунок 4 – Интерфейс приложения распознавания и синтеза речи

Словарь в созданном приложении состоит из 100 основных слов. Из этих слов и составляются все тестовые предложения, примеры и обучающие множества.

В документации к Sphinx4 сказано, что кроме фонем, используемых для распознавания слов, существуют еще 3 фонемы, которые не являются звуками: <s> - начало предложения, <sil> - тишина в середине предложения, </s> - конец предложения. По сути, все эти фонемы обозначают тишину и отличаются лишь своим положением в сказанной фразе. Система определяет их сама и пользователю совершенно не надо о них беспокоиться.

В виду того, что грамматику Pascal нет возможности приспособить для создания грамматической модели используемой в Sphinx4, из-за использования правых и левых рекурсий, было решено использовать наиболее популярную языковую модель – N-граммную (рис.5) [8-10].



\1-grams:
-0.9382 </s> -0.3010
-0.9382 <s> -0.1647
-2.8224 AND -0.2255
-2.8224 ARRAY -0.2472
-3.2204 ASM -0.2478
-2.2910 ASSIGN -0.2068
-3.5214 ASTERISK -0.2478
-3.2204 AT -0.2438
-3.5214 BACKSLASH -0.2478
-2.3173 BEGIN -0.2466
-2.7432 BOOLEAN -0.2365
-2.7432 BYTE -0.2365
.....

\2-grams:
-2.8842 <s> AND -0.2553
-2.5832 <s> ARRAY 0.0000
-2.8842 <s> ASM 0.0000
-1.8050 <s> ASSIGN 0.0000
-2.8842 <s> ASTERISK 0.0000
-2.5832 <s> AT 0.0000
.....
-0.3010 ASM </s> -0.3010
-1.2304 ASSIGN </s> -0.3010
-1.5315 ASSIGN FALSE 0.0000
-1.5315 ASSIGN LEFT-PARENTHESIS -0.0458
-1.5315 ASSIGN NIL 0.0000
-1.5315 ASSIGN NOT -0.1249
-1.5315 ASSIGN RANDOM -0.2218
-1.5315 ASSIGN SQUARE-ROOT -0.0969

\3-grams:
-0.3010 <s> AND </s>
-0.6021 <s> ARRAY </s>
-0.6021 <s> ARRAY LEFT-SQUARE-BRACKET
-0.3010 <s> ASM </s>
-1.3802 <s> ASSIGN </s>
-1.3802 <s> ASSIGN FALSE
-1.3802 <s> ASSIGN NIL
-1.3802 <s> ASSIGN NOT
-1.3802 <s> ASSIGN RANDOM
-1.3802 <s> ASSIGN SQUARE-ROOT
.....
-0.3010 <s> DO </s>
-0.6021 <s> DOT </s>
-0.6021 <s> DOT DOT
-0.6021 <s> DOUBLE </s>
-0.6021 <s> DOUBLE SEMICOLON
.....
-0.3010 COLON ARRAY </s>
-0.3010 COLON BOOLEAN SEMICOLON
-0.6021 CONSTRUCTOR NAME LEFT-PARENTHESIS
-0.6021 CONSTRUCTOR NAME SEMICOLON

Рисунок 5 – N-граммная языковая модель

### *Полученные результаты*

Для тестирования разработанных моделей были написаны различные программы на языке программирования Pascal. Пример распознаваемых аудиофайлов, описывающих речевые произношение операторов исходного текста программы изображен на рисунке 6.

Входные данные для тестирования состоят из записанных фраз и их текстового представления для того, чтобы программный продукт мог их сопоставлять в ходе тестирования.

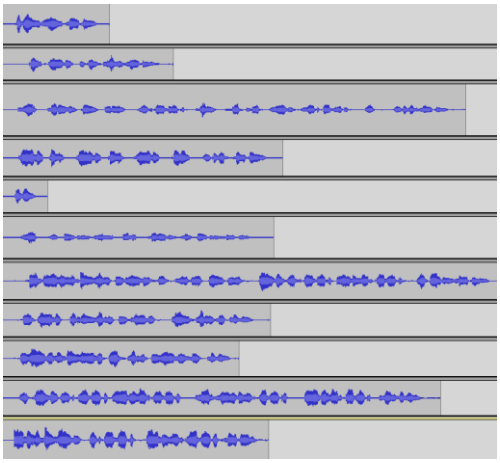


Рисунок 6 – Примеры тестовых аудиофайлов

Одним из ключевых моментов является то, что в виде тестового примера приложению необходимо подавать заново записанные примеры, а не те аудиофайлы, на которых производилось обучение системы [13], чтобы не повлиять на качество результатов тестирования.

Суть эксперимента заключалась в определении достаточного количества обучающего материала для хорошего уровня распознавания системы. Грамотно составленная языковая модель сильно влияет на распознавание речи, и она была составлена из списка самых часто встречающихся конструкций языка Pascal. Однако еще более существенный вклад в качество распознавания речи вносит акустическая модель и по важности она занимает первое место. Важно обучить модель так, чтобы она лучше понимала, что говорит человек и как вне зависимости от окружения или говорящего. В данном случае велась разработка однодикторных акустических моделей в спокойном окружении, что значительно повышает шансы на верное распознавание речи.

Всего в тестовой программе было 144 слова, и аудиозапись длилась 123,67 секунд. Каждая из акустических моделей проходила ряд тестов с использованием двух языковых моделей. Первая – модель, построенная на словах, вторая – по часто встречающимся словосочетаниям.

Первый тест проводился с акустической моделью, обученной на 100 аудиофайлах. На этой маленькой модели ясно видно, что при малом количестве обучающего материала, распознавание будет сильно зависеть от качества языковой модели. При плохой языковой модели процент ошибки составил 48,96% [12], а при хорошей – 37,44%.

Второй тест проводился с акустической моделью, обученной на 293 аудиофайлах. На этой модели также видна зависимость качества распознавания от языковой модели. При плохой модели ошибка составила 12,96%, а при хорошей – 2,88%.

Третий тест проводился с акустической моделью, обученной на 500 аудиофайлах и на этом тесте уже не так заметна разница в качестве распознавания в зависимости от различных типов моделей. При плохой модели ошибка составила 7,2 %, а при хорошей – 2,87%. Это может сказать о том, что чем больше имеется обучающего материала, тем меньше распознавание зависит от качества языковой модели.

Все результаты проведенных тестов изображены на рисунках 7 – 10.

Одним из важных аспектов распознавания речи является оценка скорости распознавания речи и количество потребляемой оперативной памяти.

Все значения на диаграмме в секундах и по ней мы можем судить, что с увеличением количества обучающих аудиозаписей увеличивается и время распознавания речи. Любое из значений на диаграмме меньше, чем время, затраченное на произнесение речи, что является хорошим показателем.

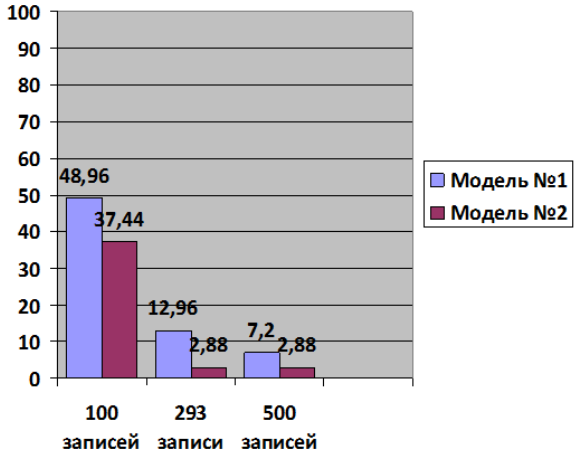



Рисунок 7 – Процент возможной ошибки распознавания

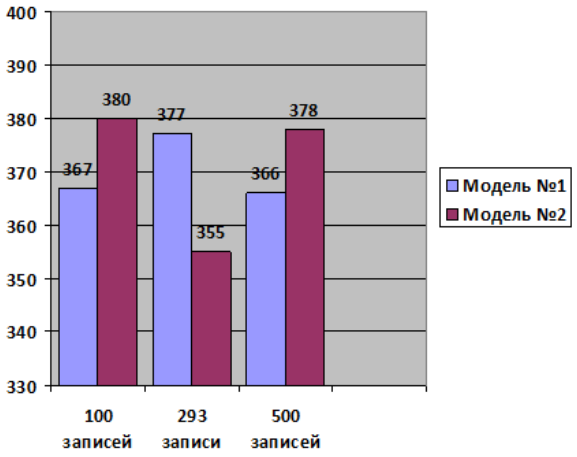



Рисунок 8 – Потребляемая приложением память в мегабайтах при распознавании речи

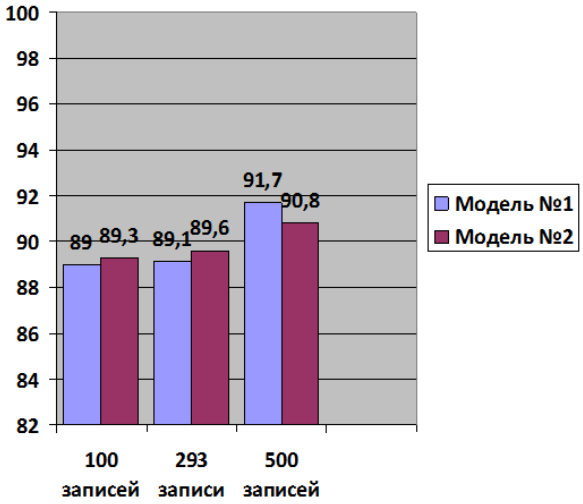


Рисунок 9 – Время распознавания моделями аудиозаписи длительностью 123,67 секунды

Количество потребляемой памяти явно не зависит от качества моделей и от количества обучающего материала. Всего приложению было выделено 400мб оперативной памяти и во всех случаях количество потребляемой разрабатываемым программным продуктом памяти приближалось к этому порогу.

### *Выводы*

Средствами системы Sphinx были разработаны языковые и акустические модели для словаря в 101 слово. В работе был выполнен анализ эффективности созданных моделей с помощью ряда тестов системы Sphinx на реальном примере программы. Тестирование показало, что данная инструментальная среда пригодна для создания приложений распознавания речи. Качество распознавания зависит от языковой модели и, главным образом, от акустической модели. Важно внимательно подбирать обучающие данные.

Данный программный продукт представляет собой однодикторное приложение для ввода и вывода текстов программ речью, а также позволяет редактировать их и отправлять на компиляцию. Приложение может послужить хорошим инструментом для облегчения взаимодействия пользователя с компьютером во время написания программ на языке Pascal.Количество потребляемой памяти явно не зависит от качества моделей и от количества обучающего материала. Всего приложению было выделено 400мб оперативной памяти и во всех случаях количество потребляемой разрабатываемым программным

### *Литература*

1. Федяев О.И., Бакаленко В.С., Савкова Д.Г. Речевой интерфейс для интеллектуализации ввода исходного кода программ: Материалы XV международной научной конференции им. Т. А. Таран «Интеллектуальный анализ информации». – Киев: КПИ, 2015. – с.250-256.

2. Федяев О.И., Бакаленко В.С. Intelligent interface for voice input of program source code: Материалы XI международной научно-технической конференции «Интерактивные системы: проблемы человеко-компьютерного взаимодействия». – Ульяновск: УГТУ, 2015. – с.120-126.

3. Бакаленко В.С., Федяев О.И. Разработка речевого распознавателя на основе моделей языка в среде CMU Sphinx: Материалы международной научно-техническая конференции студентов, аспирантов и молодых ученых «Компьютерная и программная инженерия - 2015». – Донецк: ДонНТУ, 2015. – с.34-37.

4. Синтез и распознавание речи. Современные решения [Электронный ресурс]. – Режим доступа http://www.frolov-lib.ru/books/hi/ch05.html. – Суббота 5 сентября 2015 10:00:00.

5. Dynamic Programming Algorithms in Speech Recognition / Furtuna T.F. // Revista Informatica Economica. – 2008. - №2(46). – p. 94. – Режим доступа: http://revistaie.ase.ro/content/46/s%20-%20furtuna.pdf. – Воскресенье 6 сентября 2015 16:00:00.

6. CMU Sphinx Open Source Toolkit For Speech Recognition Evaluation [Electronic resource]. – Режим доступа: http://cmusphinx.sourceforge.net. – Четверг 18 июня 2015 12:30:00.

7. Free Pascal Open source Compiler for Pascal and Object Pascal[Электронный ресурс]. – Режим доступа: http://www.freepascal.org. – Среда 23 декабря 2015 11:30:00.

8. Скрытая марковская модель [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Скрытая_марковская_модель. – Среда 1 июля 2015 22:00:00.

9. Скрытые марковские модели [Электронный ресурс]. – Режим доступа: https://ru.wikibooks.org/wiki/Скрытые_марковские_модели. – Четверг 2 июля 2015 10:00:00.

10. N-грамм [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/N-грамм. – Среда 9 сентября 2015 18:00:00.

11. Word Error Rate Calculation [Electronic resource]. – Режим доступа: http://martin-thoma.com/word-error-rate-calculation. – Среда 1 июля 2015 20:00:00.

12. Model Adaptation using MAP [Electronic resource]. – Режим доступа: http://www1.icsi.berkeley.edu/Speech/docs/HTKBook/node130_ct.html. – Пятница 16 октября 2015 18:00:00.

13. Training Acoustic Model for CMUSphinx. – Режим доступа: http://cmusphinx.sourceforge.net/wiki/tutorialam. – Среда 15 июля 2015 22:00:00.

***Bakalenko V., Fedyaev O. Speech interface for intellectualization of computer programs input.*** *The article describes the system of voice input of a program text in a programming language. Programmers seem speech interface more natural way to set a program source code. The core of interface is acoustic-linguistic model of automatic speech recognition.*

***Keywords***: *speech, Sphinx, recognition, synthesis.*

*Статья поступила в редакцию 21.05.2016*
*Рекомендована к публикации д-ром техн. наук В.Н. Павлышом*