УДК 004.9+81'322.2:811.512.133

ПОДХОДЫ СОЗДАНИЕ АЛГОРИТМА СТЕММИНГА ДЛЯ УЗБЕКСКОГО ЯЗЫКА.

Бакаев Илхом Изатович
bakayev2101@gmail.com
Учёный секретар НИИ Развития цифровых технологий и искусственного интеллекта
доктор философии по техническим наукам (PhD)
Бакаева Райхон Изатовна
raihonbakaeva@yandex.ru
Учительница Бухарского педагогического коллежа

Annonatsiya. Bu ishda lingvistik qoidalar leksikoni va o‘zbek tili uchun so‘z shakllari ma’lumotlar bazasidan foydalangan holda stemming algoritmi ishlab chiqilgan. Taklif etilayotgan algoritmning aniqligi tokenizatsiya algoritmining aniqligiga bog‘liq. Shu bilan birga, probel bilan ajratilgan qo‘shma so‘zlarning asoslarini topish masalasi bu yerda ko‘rilmagan, chunki bu muammo to‘g‘ridan-to‘g‘ri tokenizatsiya bosqichida hal qilinadi.

Аннотация. В данной работе разработан алгоритм стемминга с использование лексикона лингвистических правил и базы данных нормальных форм слов для узбекского языка. Точность предложенного алгоритма зависит от точности работы алгоритма токенизации. При этом, вопрос нахождения корней сложных слов, разделенных пробелами, здесь не рассматривался, так как эта задача решается, непосредственно, на этапе токенизации.

Abstract: In this paper, a stemming algorithm has been developed using a lexicon of linguistic rules and a database of normal word forms for the Uzbek language. The accuracy of the proposed algorithm depends on the accuracy of the tokenization algorithm. At the same time, the issue of finding the roots of compound words separated by spaces was not considered here, since this problem is solved directly at the tokenization stage.

Kalit so‘zlar: *tokenizatsiya, stemming, lemmatizatsiya, qo‘shma so‘zlar, tub va yasama so‘zlar, takroriy so‘zlar, juft so‘zlar, affiks, algoritm*

Ключевые слова: *токенизация, стемминг, лемматизация, простые слова, сложные слова, парные слова, повторяющиеся слова, аффикс,*

Keywords: *affix, repeating words, paired words, difficult words, simple words, lemmatization, stemming, tokenization, algorithm*

1. Введение. В информационном обществе доля текстовый информации значительно растёт день за днём. А это свою очередь привело к требованию создание новых методов и алгоритмов в области обработки естественного языка. На этом плане поисковые системы Google, Bing, yahoo, Yandex является передовиками разных проектах, связанных с поиском и извлечением информации, машинным переводом, проверкой орфографии слов, проверкой грамматики предложений и т.д. Хотя цель этих задач разные, но они пресекаются подзадачах стемминга и лемматизации. Стемминг и лемматизация является одним из типов морфологического анализа, который широко используется в информационном поиске.

Стемминг – это алгоритм для определения основы словоформы. А лемматизация – приведение словоформы в словарные формы, который называется леммой. На сегодняшний день существует множества стеммеров и лемматизаторов (реализованный алгоритми) как AOT, mystem, TreeTagger, pymorphy2, CrossMorphy, ARME, Cir_morph, Crosslator и др., который предназначены для флективных и частности агглютинативных языков. Но основная проблема применение этих алгоритмов в том что они не универсальны и требуют особого подхода модификации с учетом структуры слов для каждого языка. К этим языкам относится тюркский, казахский, татарский, киргизский, туркменский, узбекские языки, который входит в семью агглютинативных языков. За последние время активно разрабатываются множество модификации этих алгоритмов. Ниже приведено некоторые из них для разных языков.

2. Связанные работы.

Авторы [Razmi и др., 2021, с. 365] сравнивают алгоритмов стемминга Портера и Ланкастера для английского языка. Алгоритм оценивается на основе ошибок результата стемминга и визуализации текстовых данных. Качестве ресурса используется 10 текстовых документов, который выбрано случайно из газет интернета. Выбранные документы касается в сферы цен нефтяных продуктов и технология блокчейна. Приведённые результаты показывают, что алгоритм Портера работать 43% лучше, чем алгоритм Ланкастера.

Авторы [Rakhimova, Turganbaeva, 2020, с. 545] описывают алгоритм нормализация слов на основе классификации окончаний и суффиксов. Приведены все возможные варианты суффиксов и окончаний, общее количество которых составляет 26526 суффиксов и 3565 окончаний. Построен

конечный автомат для всех частей речи с учетом морфологическими свойствами казахского языка. А также разработан алгоритм стемминга без словаря на основе классификации суффиксов и окончаний. Алгоритм протестирован на текстовых корпусах казахского языка.

В статье [Sharma, Kumar, Mansotra, 2016, с. 11449] предлагается алгоритм стемминга для индийского языка. Алгоритм основывается на гибридный подход, который включает brute force, удаление и замена суффиксов. Алгоритм состоит из нескольких шагов: 1) удаляется стоп-слова, 2) осуществляется поиск по базе данных с использованием метода brute force. Если в базе данных слово существует, тогда выводится строка связанный с этим словом. Но если слово в базе данных отсутствует, тогда суффиксы удаляется из исходного слова. После удаление суффикса данное слово присутствует в база данных то выводится строка связанный с этим словом. Если слова снова отсутствуют, то подход суффиксной замены применяется везде, где это необходимо. Результат исследование дал 92 % точность для имен существительных индийского языка.

В этом исследование [Winarti и др., 2017, с. 1758] описан стеммер для индонезийского языка. Разработанный алгоритм под называнием CAT основывается на морфологических правилах языка. Алгоритм состоит из трех компонентов, таких как группировка аффиксов, приоритет морфологических правил и словарь корней слов. Все правила, включенные в алгоритмы проверены и удачные результаты были сохранены в листе кандидатов.

В работе [A. Sharifloo, Shamsfard, 2008, с. 583] предложен алгоритм основанных морфологических правил, который работает по принципу снизу в вверх. Алгоритм состоит из трех этапов: под строковые теги, соответствие правил, соответствие антиправил. Этот алгоритм включает 252 правил и 20 антиправил. На первом этапе извлекаются морфологические свойства для всех возможных подстрок слово. Определяется морфемы и их кластеры, который они принадлежать, а также ядро (любой основа слово) слов. На этапе соответствие правил для каждого ядра извлекается соответствующие правила. Не подходящие ядро к правилам переходить на следующий этап с соответствие антиправил. Алгоритм оценивается с помощью корпуса «Hamshahri» , который извлекает 90,1% правильных основ слов.

Основной недостаток рассмотренных выше алгоритмов стемминга состоит в том, что найденные с их помощью основы, далеко не всегда соответствуют морфологическому основу (корню) слов. Обычно такие

проблемы возникают в тех случаях, когда тип слова заранее неизвестен, либо словоформа образована в соответствии с несколькими правилами. Таким образом, происходит ситуация, выделенные основы слов сами по себе не имеют смысла.

Целью данной работы является разработка метода нормализации текстов на узбекском языке на основе стемминг алгоритма для обработки слов, структурный тип которых известен заранее.

3. Постановки задачи. Морфологический состав слов узбекского языка включает двух основных частей: корневых и аффиксальных морфем. Корень (o‘zak) – это морфема, которая имеет лексическое значение и не имеет аффиксов в своём составе. Аффикс – это морфема, которая может иметь несколько функций. Присоединяясь к корню аффикс образует новое слово или грамматические значение. Нормальная форма (основа) слов в узбекском языке (negiz) может состоять только из корневой морфемы либо из корневой и аффиксальной морфем одновременно.

Основа слов может быть производной и непроизводной. Производная основа состоит из корня и словообразующего аффикса. Например, «gul» (цветок) является непроизводной основой, «guldon» (ваза) – производная основа. Словообразование может происходить различными способами, в том числе, путем присоединения аффикса к корню «gul+chi» (флорист), путем конкатенации двух простых слов через дефис «baxt-saodat» (счастье), «tez-tez» (часто). Таким образом, в современном узбекском языке слова по своей структуре бывают: простые, сложные, парные, повторяющиеся (рис. 1).

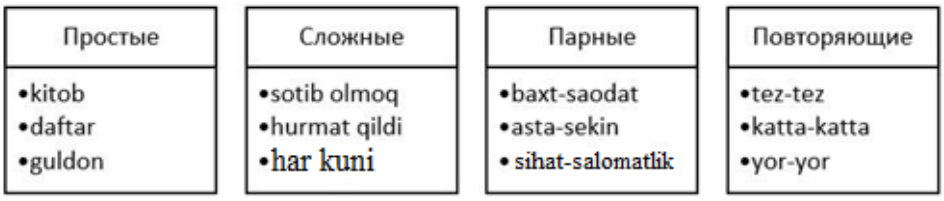

Рис.1. Типы слов по структуре

Отметим, что соответствующее программное средство для определения типов слов узбекского языка, основанное на применении регулярных выражений, было создано нами ранее.

3. Метод решения. Разработанный алгоритм варьируется в зависимости от типов слов, которым соответствуют следующие обозначения: w_s – простые слова (с производной или непроизводной основой); w_c - – сложные слова; w_p– парные слова; w_r – повторяющие слова. Наибольший интерес представляет

задача нормализации слов типа w_p и w_r, например, соответственно «sihat-salomatlik» и «tez-tez». Суть алгоритма сводится к тому, что из словоформы удаляются аффиксы, встречающиеся после знака дефис. Если исходное слово не имеет аффиксов, то оно считается стеммой. То есть, «sihat-salomatlik» усекается до «sihat-salomat» (здравствующий), «tez-tez» это сам стемма (рис. 2). Далее опишем переменные и функции алгоритма «StemWCWP» для типа w_p : «S» – строка; «Spart» – строка, содержащая часть строки S; «b» – логическая метка; «i» – Итерационная переменная; «HyphenAfterStr(S)» – пользовательская функция, вырезающая часть строки после знака дефиса; «Length(string S)» - стандарная функция возвращения длины строки; «dbStemShow(string S)» – пользовательская функция, которая проводит поиск по базе данных; «SubStr(firstindex:int, lastindex:int)» – стандарная функция, извлекающая символы, начиная с индекса firstindex до индекса lastindex;

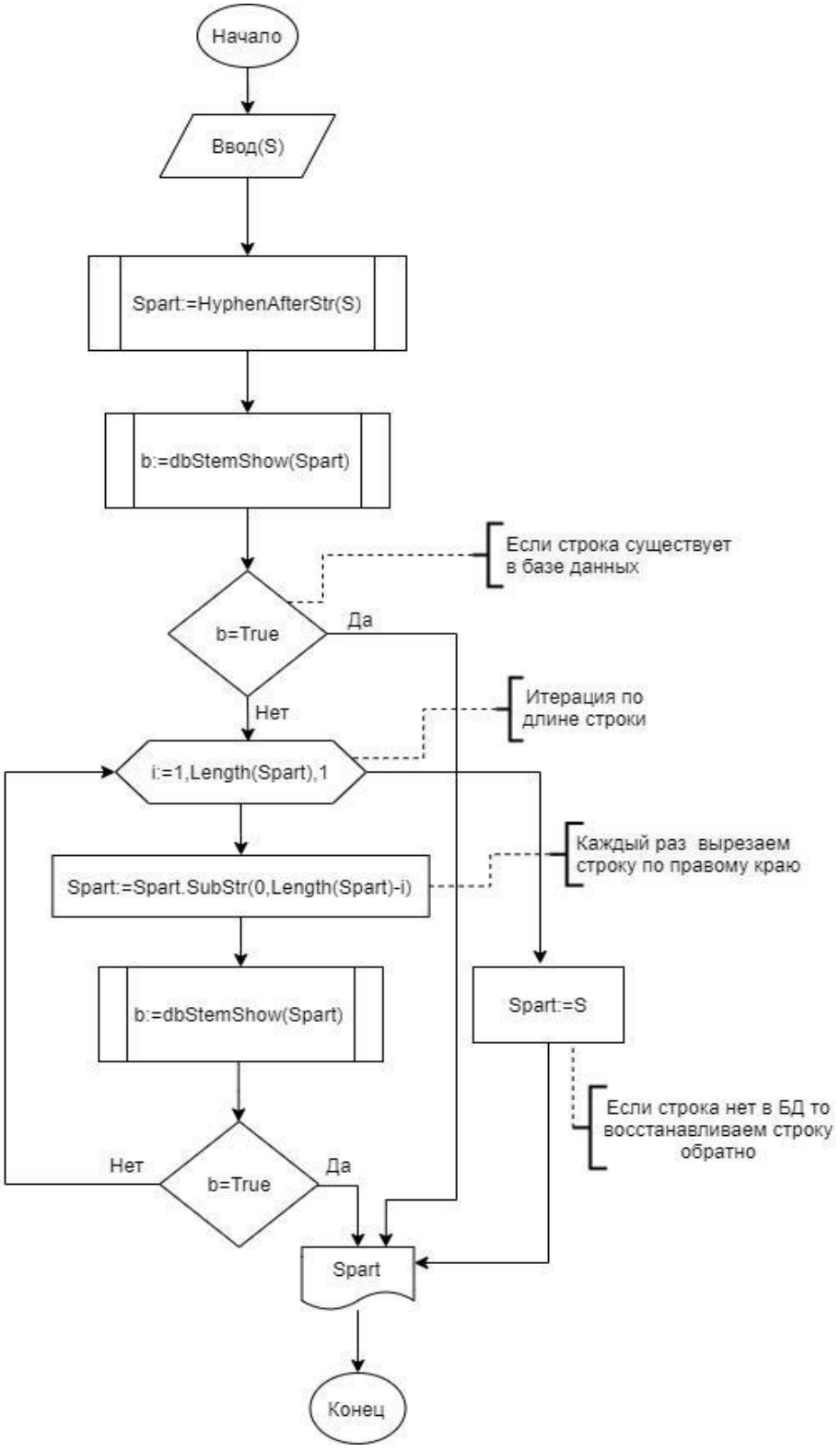

Рис.2. Блок схема алгоритма StemWCWP

Слова типа w_p - «последовательность символов алфавита + знак дефис + последовательность символов алфавита», где слова, разделённые дефисом,

могут быть идентичными либо первое слово может быть подстрокой второго слова. Во втором случае из второго слова отсекаются любые аффиксы (рис.3).

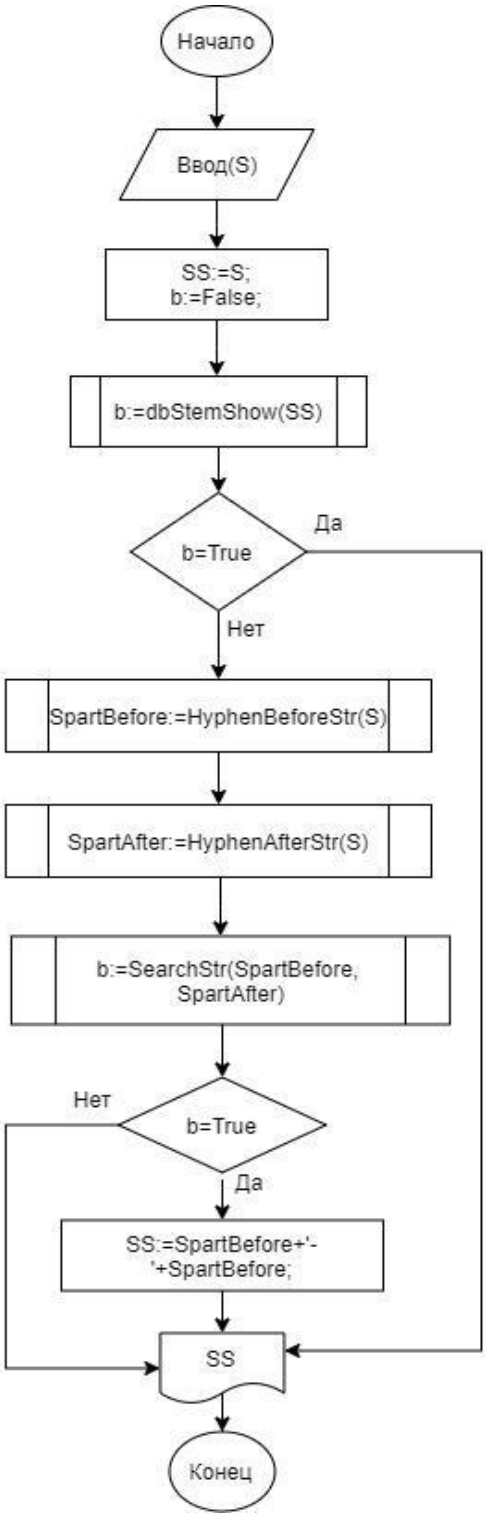

Рис.3. Блок схема алгоритма StemWP

Описание переменных и функций алгоритма StemWP: - «S» – строка; «SS» – строка, содержащая дубликат строки S; «b» – логическая метка; «SpartBefore» – содержит результат функции HyphenBeforeStr (string S); «SpartAfter» – содержит результат функции «HyphenAfterStr(string S)»; «HyphenBeforeStr (string S)» – пользовательская функция, вырезающая часть строки до знака дефиса; «HyphenAfterStr(string S)» – пользовательская функция, вырезающая часть строки после знака дефиса; «SearchStr(string str1, string str2)» – пользовательская функция, проверяющая является ли "str2" частью "str1", если да то возвращает значение true, в противном случае - false. «dbStemShow(string S)» – пользовательская функция, которая проводит поиск по базе данных;

Слова типа w_s состоят из «последовательность символов алфавита». Иногда в этих словах встречается знак апострофа. Например, «a’lo». Чтобы найти основу для такого типа слов удаляется префикс или аффиксы. (рис.4).

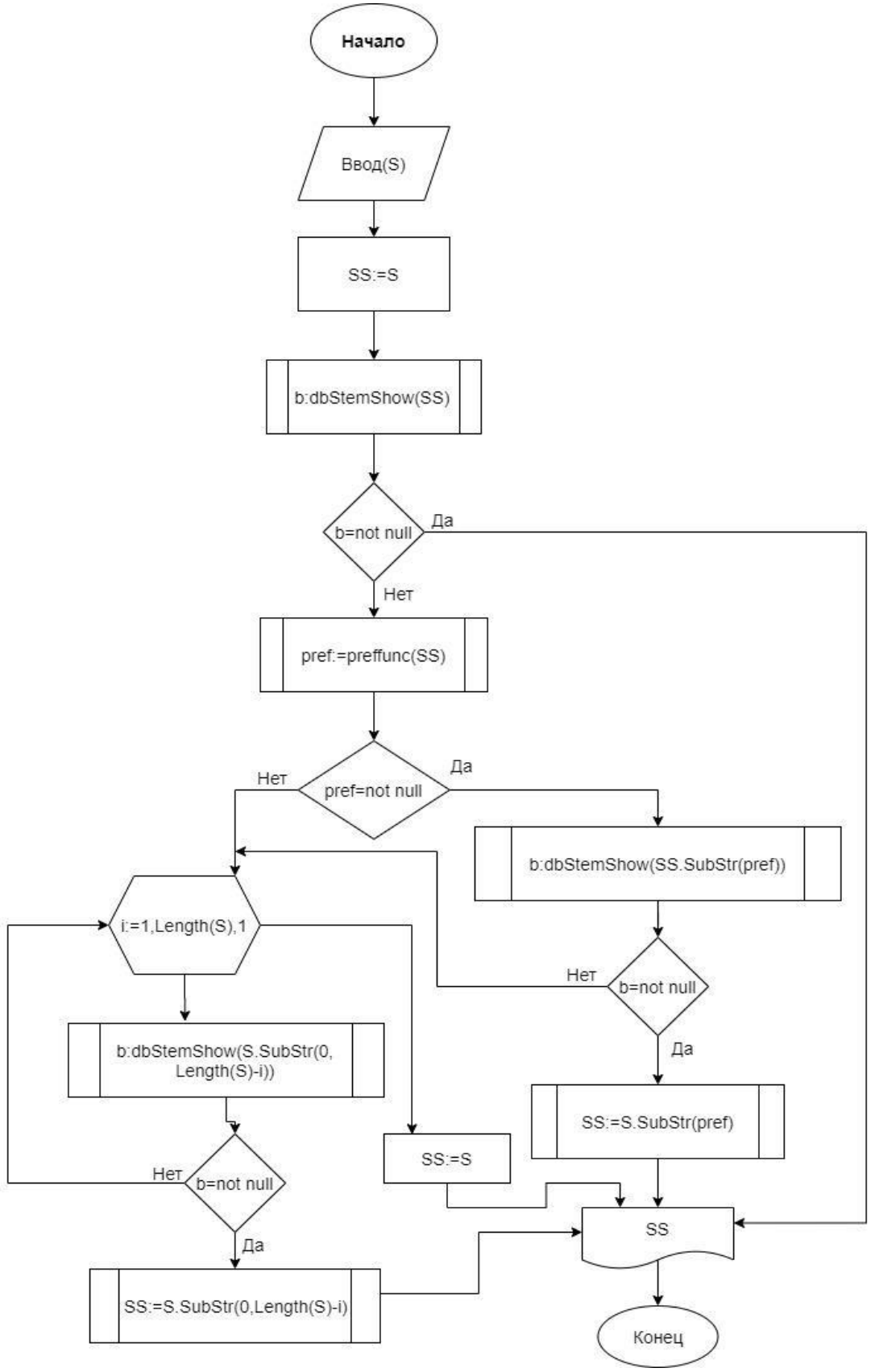

Рис.4. Блок схема алгоритма StemWS

Далее опишем переменные и функции алгоритма StemWS - «S» – строка; «SS» – строка, содержащая дубликат строки S; «b» – логическая метка; - «i» – итерационная переменная; «pref» – строка для префиксов; «Length (string S)» – стандартная функция, возвращающая длину строки; «dbStemShow(string S)» – пользовательская функция, которая проводит поиск по базе данных; «preffunc(string S)» – пользовательская функция, извлекающая префиксы из строковой переменной; «SubStr(firstindex:int, lastindex:int)» – стандартная функция, извлекающая символы, начиная с индекса firstindex до индекса lastindex;

Далее опишем переменные и функции алгоритма StemWS: «S» – строка; «SS» – строка, содержащая дубликат строки S; «b» – логическая метка; «i» – итерационная переменная; «pref» – строка для префиксов; - «Length (string S)» – стандартная функция, возвращающая длину строки; «dbStemShow(string S)» – пользовательская функция, которая проводит поиск по базе данных; «preffunc(string S)» – пользовательская функция, извлекающая префиксы из строковой переменной; «SubStr(firstindex:int, lastindex:int)» – стандарная функция, извлекающая символы, начиная с индекса firstindex до индекса lastindex;

Обсуждение результатов. Для тестирования алгоритма использовался текстовые корпуса по жанрам: законодательные документы, сказки, узбекские пословицы и религиозные произведения табл.1.

4. Обсуждение результатов. Для тестирования алгоритма использовался текстовые корпуса по жанрам: законодательные документы, сказки, узбекские пословицы и религиозные произведения табл.1.

Таблица №1. Результати алгоритма стемминга.

№	Выбранные текстовые корпуса по жанрам	Число слов	Правильные основы
1.	Сказки	1246	96%
2.	Религиозные произведения	1285	91,3%
3.	Узбекские пословицы	1200	94%

5. Заключение. Таким образом, разработан метод нормализации текстов на узбекском языке на основе алгоритма стемминга. При разработке алгоритма

использовался гибридный подход на основе совместного применения алгоритмического метода, лексикона лингвистических правил и базы данных нормальных форм слов узбекского языка. Точность предложенного алгоритма зависит от точности работы алгоритма токенизации. При этом, вопрос нахождения корней сложных слов, разделенных пробелами здесь не рассматривался, так как эта задача решается, непосредственно, на этапе токенизации[Bakaev I.,2021, с.1]. Алгоритм может быть интегрирован в различные автоматизированные системы машинного перевода, извлечения информации, информационного поиска и др.

Литературы

1. A. Sharifloo A., Shamsfard M. A Bottom Up approach to Persian Stemming // IJCNLP. , 2008. С. 583–588.

2. Rakhimova D. R., Turganbaeva A. O. Normalization of kazakh language words // Sci. Tech. J. Inf. Technol. Mech. Opt. 2020. Т. 20. № 4. С. 545–551.

3. Razmi N. A. и др. Visualizing stemming techniques on online news articles text analytics // Bulletin of Electrical Engineering and Informatics. , 2021.

4. Sharma A., Kumar R., Mansotra V. Proposed Stemming Algorithm for Hindi Information Retrieval // Int. J. Innov. Res. Comput. Commun. Eng. (An ISO Certif. Organ. 2016. Т. 4. № 6. С. 11449–11455.

5. Winarti T. и др. Improving Stemming Algorithm Using Morphological Rules // Int. Adv. Sci. Eng. Inf. Technol. 2017. Т. 7. № 5. С. 1758–1764.

6. Bakaev I.I. Linguistic features tokenization of text corpora of the Uzbek language // Bulletin of TUIT: Management and Communication Technologies. – 2021. Vol. 4. – P. 1-8.