Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика
Катедра „Компютърна Информатика“

ДИПЛОМНА РАБОТА
на тема

Разширение за текстови редактори за откриване и контекстно-зависима замяна на чуждици

Дипломант:
Ангел Георгиев, специалност: „Информатика“, магистърска програма „Изкуствен Интелект“, факултетен № 26098

Научен ръководител:
проф. д-р Иван Койчев, катедра „Софтуерни технологии”, ФМИ, СУ „Св. Климент Охридски”

Консултант:
ас. Борис Величков, катедра „Софтуерни технологии ”, ФМИ, СУ „Св. Климент Охридски"

София, 2021 година

Съдържание

1 Увод

1.1 Мотивация за работа

Навлизането на чужди думи в българския език е процес, който трае от векове. Използването на този тип изразяване не прави впечатление на повечето хора в нормалното ежедневие, но то често пъти може да доведе до проблеми и неразбиране между общуващите.

Чуждица, чужда дума и заемка са три често използвани фрази от комуникиращите страни, които не винаги биват разбрани от хората. Чуждите думи в българския език са думите, заети от чужди езици, които навлизат или вече са навлезли в местния. Те са обект на изследване от емпрунтологията, наука за проучване на заемането на думи от чужд произход в даден език [1]. Някои от тези са думите: *шпионаж, палачинка, мерси, чадър, фризьор, кайт*. Посочените примери са навлезли в битието ни отдавна и не правят впечатление на общуващите, но за разлика от тях има други, които могат да бъдат доста неприятни или неразбираеми като *файда, лонгборд и ънфрендвам*.

Чуждите думи имат две подразделения – чуждици и заемки. Заемки[1] са думи, заети от чужд език, обикновено за назоваване на ново явление или понятие от реалността. За тези фрази нямаме български еквивалент. Примери за това са: *смартфон, сайт, смути* и други. Често тези части на речта характеризират технологични иновации или културни новости.

Чуждици са тези думи от чужд език, за които имаме думи в родния – *юзър(потребител), лайк(харесване), шеър(споделяне)*. Този тип изразяване има възможност да бъде направено с българските думи, но вместо това се избира другия вариант. Други примери за това използване на думи са *фешън, сънорт, лайфстайл, плеймейтка*. Те са описани в статията "*10-те най-често използвани чуждици и техните български прабаби"* на Марина Делева [2].

1.2 Как използването на чуждици може да има негативен ефект върху езика?

Навлизането на чужди думи в езика ни на пръв поглед изглежда като обогатяване, но това не винаги е така. Доста често използването на думите е

[1] Заемки - https://rechnik.chitanka.info/w/%D0%B7%D0%B0%D0%B5%D0%BC%D0%BA%D0%B0

неуместно и създава неадекватно звучене. Следното изречение илюстрира точно това:

Пърформансът на ***девелъпърите*** е повече от добър.

Тук думите „пърформънс“ и „девелъпър“ са чуждици, които могат да се заместят съответно с българските думи „представяне“ и „разработчик“. По този начин четимостта на изречението се вдига, без смисълът да бъде променян. Също така това спомага за по-лесното разбиране на израза.

Съществува и друг, по-голям проблем. Той се изразява в това, че доста често може да има неразбирателство от страна на отсрещната страна [3] . Това се случва при използването на стари думи (т.нар архаизми) или диалектни думи (например *авджия, бошем, кошуля*), които младите хора не са чували и не подлежат на разбиране от тях. Има и думи навлезли директно от английския (например *шервам*, хоствам, стриймвам). Да разгледаме следните примери:

Младите хора вече не се интересуват от ***авджийство.***

Стрийма ми в Туич беше успешен.

Генерирането на качествен ***контент*** е важно за всеки един писател на статии.

Ъплоудването на файла е доста бавно днес.

Първите два примера биха създали дискомфорт и неразбиране у хора, които са по-млади или не са имали досег с края, от който са произлезли думите.

Останалите изречения пък могат да бъдат непонятни за голяма част от хората с по-ниска компютърна грамотност, а други биха били подразнени, защото съответните думи (*стрийм, генериране, контент, ъплоуд*) си имат български заместители:

Живото ми предаване в Туич беше успешен .

Създаването на качествен ***съдържание*** е важно за всеки един писател на статии.

Качването на файла е бавно днес.

1.3 Основни фактори определящи употребата на чужди думи

През последните десетина години българският език е залят от чужди думи, заместващи дори понятия, с които родния език разполага. Старши научен сътрудник Сия Колковска от института по български език при БАН е

категорична, че това се дължи по-скоро на ниската езикова култура и опитите да бъдеш модерен, отколкото да запълнят празнина в езика [4].

1.3.1 Доколко ни прави впечатление наличието на чужди думи?

Все повече чужди думи навлизат в речта ни. Основно те са английски, свързани са с новите технологии и с глобалната информационна мрежа и трудно могат да намерят български съответствия.

Десетки са примерите за нови думи, които вече са част от езика ни в последните години.

От Института по български език на БАН предложиха в интернет страницата си хората да измислят български съответствия на някои от тях. Например дрон и предложението „безлетец", което е събрало най-много харесвания, флашмоб – „бързосбор", и имиджмейкър – „ликотворец". Българските варианти едва ли ще станат част от речника, но заемките твърде вероятно, съобщава БТВ.

„Туитвам", „лайквам", „шервам" – тези чуждици например може да намерят място в речника на новите думи в българския език. Това е така, защото те имат широка употреба [5].

1.3.2Навлизане на чуждите думи в българския език

Около 9000 думи и съчетания са се появили в речта ни от последното десетилетие на 20-и век до наши дни. 5000 от тях пък са възникнали след 2000та година, по думите на проф. Диана Благоева-Стефанова от Института за български език към БАН, която заяви това пред „Монитор" [6]. Тези заключения се правят на базата на данни от три речника на новите думи – издадени през 2001, 2010 и 2021 година.

За да бъде част от един такъв речник, дадена дума трябва да отговаря на няколко условия. Първото е хронологичната новост. Например, за да попадне в речника за 21 век думата трябва да е използвана за първи път в последните 20 години.

Друга характеристика е думата да има значение, което досега не е било използвано и не е имало приложение в нашето ежедневие и бит. Някои от думите, произлезли от екзотични езици като „кимчи" (продукт от традиционната корейска кухня) или „амигуруми" (вид японско изкуство) означават явления, типични за съответното общество, от което произхождат, но са напълно нови за българския език. Тези названия се наричат още и неологизми[2].

Подобни тенденции показват думи, образувани спрямо словообразувателните закони на езика. В доста редки случаи изрази като наскоро навлезналия "ваканцувам" да станат употребими от обществото, смята експертът по езици – професор Стефанова.

[2] Тълковен речник - http://rechnik.info/%D0%BD%D0%B5%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B7%D1%8A%D0%BC

Преди съответна чужда дума да започне да се употребява в българския език като официална такава, трябва да се провери дали тя не води до появата на нови такива. В статията [6] се дава пример с думата “инфлуенсър”, около която се образува обръч от новосформиращи се слова, производни на нея – “инфлуенсърка“, „инфлуенсърство“, „инфлуенсърски“ и „инфлуенсърче“. Този начин да се назове даденото занимание може да остане в българския език, освен ако не се намери адекватно съответствие.

1.3.3Източници на чуждите думи

От въведението разбрахме, че голяма част от чуждиците и заемките навлизат от английския език, поради наличието на различни фактори, влияещи върху формирането на лексиката. Причина за това са технологичния свят, който ни заобикаля, прекараното време в социалните мрежи, както и глобализацията, която се осъществява. Освен това голяма част от навлезлите думи идват и от турски език, останали още от Османско време: аферим (браво), авер (другар), барабар (заедно) и други [7]. Има и други езици, от които сме взаимствали слова [8]:

- английски - *симпозиум*, *митология*, *динамика*, *департамент*, *буржоазия*
- немски – *клапа*, *клапа*, *бленда, багер, щепсел*
- италиански - *терца*, *опера*, *тенор*, *сопрано*, *модерато*, *мандолина*
- френски- *гарнизон*, *ескадрила*, *интендант*, *ешалон*

1.3.4Защо чуждите думи се употребяват толкова често ?

Съществува известна разлика между използването на заемки и чуждици.При наличието на първия тип слова изборът на човека се ограничава, защото както знаем заемките нямат български еквивалент и това прави включването им в речта задължително до известна степен. Когато се говори за чуждици обаче ситуацията е малко по-различна. Често причината, поради която общуването включва чужди думи е, защото комуникиращият иска да звучи по-истънчен или интелигентен, като понякога това може да му изиграе лоша шега и той да бъде неразбран. С течение на времето навлизането на дадена чужда дума става все по-силно, тъй като употребата й става все по-честа от нейните ползватели. Също така чуждиците се вливат и в жаргона на дадена група или общност. Те в доста случаи се използват като термини и тогава замяната им би била неточна и би нарушила целостта и смисъла на прилагането й.

Напливат от чуждици и чужди думи в българския език се увеличава, а това води до риск от запълване на речта с “ненужни” изрази. Въвеждането на нови слова, налага спиране на използването на по-стари думи, които с времето придобиват статут на архаизми и стават все по-неразрибаеми за новите поколения.

1.4 Цел на дипломната работа

Целта на дипломната работа е да се разработи разширение за текстови редактори, което да предлага заместващи думи (синоними) за съответните чуждици, чрез което да се улесни писмената комуникация между поколенията и да се спомогне “изчистването” на езика от ненужните думи и фрази, навлезнали от чужди езици.

Основната функционалност на приложението ще е да получава текст на български език, а като изход ще показва чуждиците, тяхното значение и предложенията за заместване. В този софтуер ще трябва да се вземе предвид и контекста, в който е попаднала дадената дума. Резултатът ще трябва да е подредени синоними спрямо най-уместното заместване, което може да бъде предложено.

Разширението ще има няколко функции. Първата ще се използва при съчиняването на даден текст. При това приложение ползвателят ще има възможността чрез удобен потребителски интерфейс да замени въпросната чуждица с подходящ синоним избран от него или в зависимост от настройката с първия в подреждането по контекст. Това ще улесни писателите да заменят ненужните думи с по-подходящи заместители и да подобрят четимостта и смисъла на изреченията.

Втората функция е по-скоро обратна като тя се изразява в това дадено количество информация да бъде анализирано при четене от потребител и да му се предоставят нужните пояснение и синоними на изписаните чуждици. Това би бил добър пример при използването на разширението от по-младите поколения, когато се опитват да обработят по-стар текст или дори когато по-възрастен човек използва системата, за да си помогне с разбирането на информация, разписана от младеж.

1.5 Задачи, произтичащи от целта

Задачите, които произтичат от целта са следните:

1. Изследване на употребата на чуждици - същност на проблема и начини за решаването му.

2. Проучване на съществуващи разработки и системи за решаването на поставената задача и подобни на нея.
3. Избор на програмни средства – среда за разработка, език, пакети, библиотеки и функции.
4. Събиране на анотирани данни и корпус от чуждици. Предварителна обработка на събраните данни.
5. Създаване на алгоритъм за намиране на най-точната заместваща дума на дадена чуждица.
6. Проектиране и реализиране на добавката в съответната среда. Тестване на разработката с наличните данни.
7. Планиране и провеждане на експерименти за оценка на представянето на системата.
8. Оформяне на дипломната работа.

1.6 Структура на дипломната работа

Структурата на дипломната работа ще има за шаблон описаните в точка 1.5 задачи, произтичащи от целта, които играят ролята на ориентири в разработката. В точка 1 е описано въведението в предметната област, която се разглежда в научния труд. Тук са указани и целите, които трябва да се изпълнят от дипломната работа. Във втората точна ще се разгледат съществуващи системи, които да решават същия или подобен проблем и възможни алгоритми, които могат да спомогнат за текущата работа. Третата точка ще показва езиковите ресурси и научни трудове, използвани за постигане на плана, описан по-горе. В точка 4 ще се покаже детайлно описание на подхода, предприет в конкретния случай. В точка 5 ще се анализират резултатите спрямо направените експерименти и създадените метрики. В точка 6 ще се завърши със заключение, а накрая ще е ситуирана библиографията в точка 7.

2 Преглед на областта

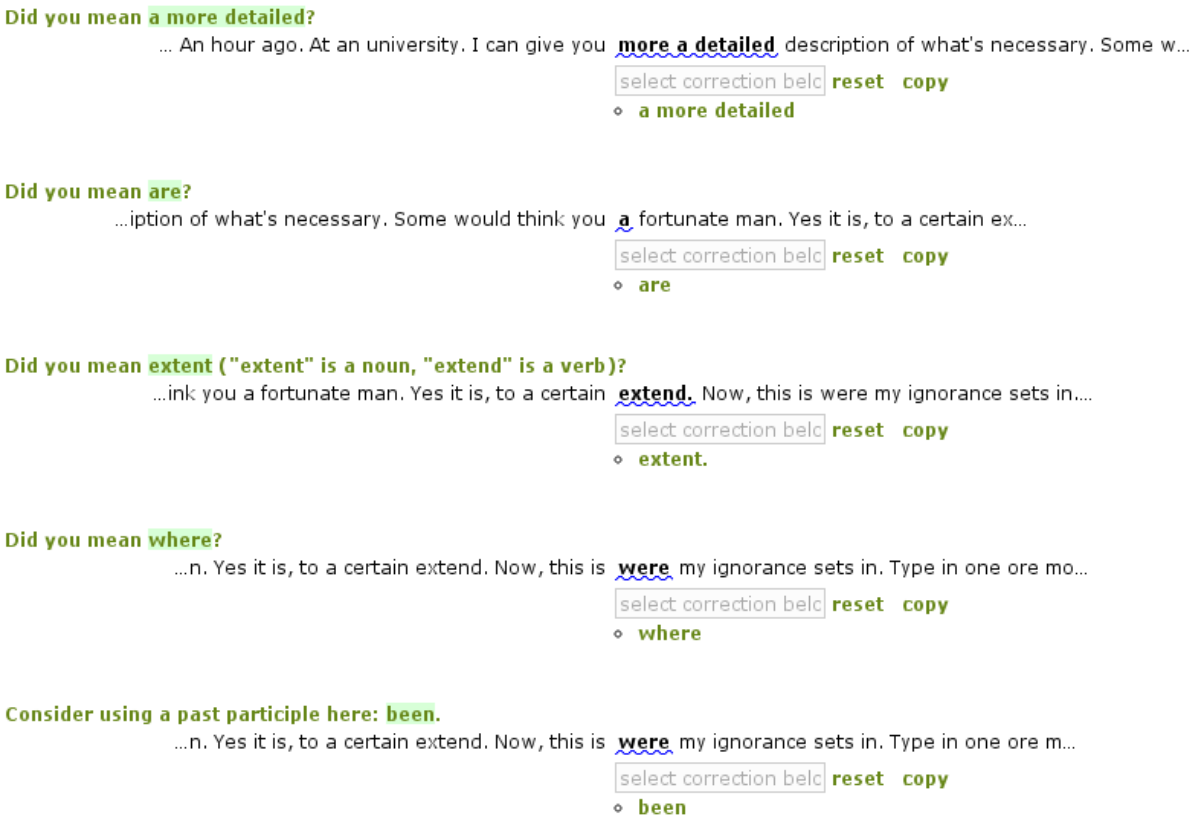
2.1 Съществуващи подобни разработки

След направеното нужно проучване успех да намеря една система, която да извършва до голяма степен работата, за която е предназначено разширението за текстови редактори и няколко системи със сходни характеристики.

2.1.1LanguageTool

LanguangeTool[3] е безплатен софтуер с отворен код, който се използва за проверка на граматиката на даден текст. Размерът на разработката е 156 мегабайта, като всички данни, с които работят n-грамите са около 8 гигабайта. Кодът може да бъде намерен в репозиторито в Гитхъб https://github.com/languagetool-org/languagetool. Тук всеки, който иска да допринесе може да разшири функционалностите, които са разписани до момента. Всичките услуги, които предлага софтуера са свободни да бъдат изтеглени от интернет. Сайтът на инструментът се свързва към себеподобен проект - LanguageTool Plus, който предоставя подобрен метод за прихващане на грешки на английски и немски език както и оптимизация при работа с по-големи тестове. Първоначално разработката е стартирала през 2003 като дипломна работа написана на езика Python от Даниел Набер. Сега платформата поддържа 31 езика, като главно са опусани от доброволци, за които съответните езици са майчини. Доста от езиците използват за изграждане на алгоритмите си N-грами [23], които ще бъдат описани по-детайлно в следващата точка. Основното приложение има възможност да бъде изтеглено за офлайн ползване, но в интернет пространството циркулира и уеб версия. Разработката може да бъде ползвана като добавка към други софтуери от типа: Microsoft Office, LibreOffice, Apache OpenOffice, Vim, Emacs, Firefox, Thunderbird, and Google Chrome. Уеб приложението има функционалност, която му позволява да бъде интегрирано в различни на вид уеб сайтове.

[3] Daniel Naber and Marcin Miłkowski - LanguageTool. https://github.com/languagetool-org/languagetool



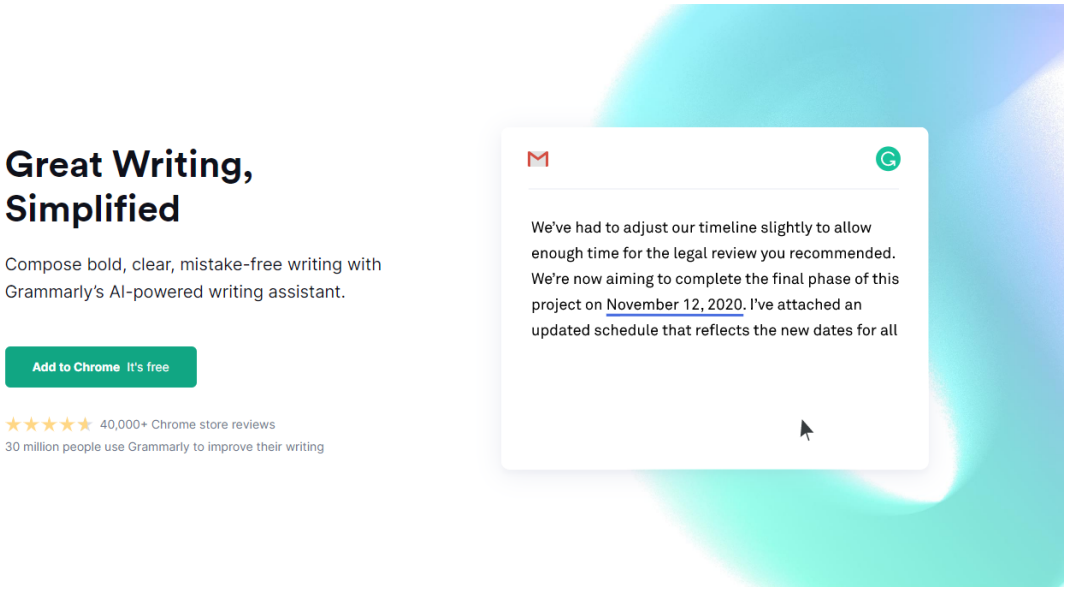
Има и други софтуери, които изпълняват подобна или същата роля като показаните по-горе. Някои от тях са Comparison of anti-plagiarism software, Grammar checker, OpenTaal, Autocorrection, Twinword и др. Голяма част от тях са платени други са безплатни, но никой от тях не изплънява нужната ефективност, която ще постигне разширението за текстови редактори. Тази бъдеща разработка ще има за цел да преобрази разписването на различни текстове със своите методи и алгоритми като придаде гъвкавост на създаването на съдържание.

2.1.2Grammarly

Grammarly[4] е асистент, помагащ писането на различни текстове, който е базиран на изкуствен интелект. Той служи за проверка на граматика, пунктоация и лексика в реално време, като уведомява потребителя за грешките по удобен начин. Методите и алгоритмите, които се използват не са опоменати, но е известно, че системата е облачно-базирана технология. Използвайки подходи от машинното самообучение програмата преглежда изречения и дори цели текстове, като се опитва да предостави възможно най-удачен вариант за заменяне на направените грешки. Този инструмент също така взема предвид и контекста на даденото съдържание. Тези функционалности са част и от някои от

[4] Alex Shevchenko, Max Lytvyn, and Dmytro Lider - Grammarly. https://www.grammarly.com/

характеристиките, които ние целим да постигнем.



2.1.3 Система за подпомагане на замяната на чуждици с подходящи синоними в текст на български

Системата за подпомагане на замяната на чуждици с подходящи синоними в текст на български (*СЗЧТБЕ*) [9] на Павел Тодоров работи с текстове на български език. Тя има две основни функционалности подобни на разрширението, които се изразяват в предлагане на синоними за заместване на чуждици и в изкарването на чуждиците по подобие на бележките под линия в книгите със съответните обяснения. Това е една много добра работа на софтуерния разработчик, в която се взима предвид контекста на дадено изречение и предлагането на думи става на базата на него. В неговата програма се използват вграждания на думи, като са разгледани възможности за N-грами и BERT алгоритъм. Това са едни от отличителните черти на въпросната система, спомагаща много читатели и писатели.

2.2 Възможни подходи за разработка

Задачата, която ще спомага за решаването на проблема, касаещ предлагането на заместител на чуждицата, който да бъде подходящ синоним е много специфична и тук се изразява в две неща, които се събират в едно – изграждането на разширение (приложение), което да решава тази задача.

Точно поради тази причина текущата точка може да бъде разделена на две подточки – едната, която се занимава със самата технология по генериране на синоними и алгоритмите, които ще разпознават чуждиците и ще предлагат

съответните им заместители и другата част на програмата, която ще е всъщност по-функционалната такава – интерфейса на самата разработка, частта, която извиква нужните команди, за да се направят съответните изчисления и да бъде препоръчана най-удачната дума или израз. Това разделение е нужно, за да се придаде абстрактност на самата документация и да може по-лесно да се разграничат подходите при изграждането на отделните слоеве на софтуера. Чрез тази детайлност ще се постигне по-голяма яснота около архитектурата, която се явява двигател на тази текстогенерираща идея.

В първата част на точката ще се разгледат възможните алгоритми за обработка на текст и прихващане на чуждици, а във втората софтуерните стъпки, които трябва да се предприемат, за да се изгради интерфейса на програмата.

2.2.1Възможни алгоритми

2.2.1.1 *Предлагане на заместващ синоним от речник*

2.2.1.1.1 Описание на алгоритъма

Използването на речник, съдържащ синоними е сравнително бърз и лесен начин да се намери предложение, което да играе ролята на чуждицата в даден речник. [9] Речникът на чуждите думи в българския език [10] би изиграл добра роля на източник на информация, тъй като той съдържа над 60000 думи. Същността на алгоритъма е проста: намира се синоним от речника, който е подходящ за подмяна на целевата дума, но самият контекст тук не можа да се има предвид. При подреждането на различни предложения за преобразуване на изречението може да се използва реда, по който са описани думите в речника.

2.2.1.1.2 Предимства и недостатъци на алгоритъма

Алгоритъмът, използващ речника с чуждите думи има няколко важни предимства. Най-голямото му предимство е бързодействието, което той постига. Простотата на самата разработка, изразяваща се в това, че речникът ще се съдържа в лесно достъпен пазител на данни като например: файл или подходяща структура в база от данни. За по-добра ефективност винаги съществува вариантът, който да бъдат използвани и така наречените InMemory операции, при които нужната работна информация (в случая речникът) бива “издърпана” в паметта, при инициализиране или стартиране на приложението, тъй като обемът на данните е сравнително малък. Този вариант на работа може да има добър ефект, само ако броят на синонимите на дадена дума е малък, тъй като тогава контекстът няма да играе голяма роля.

Както можем да забележим обаче всяка една произволна думи има значителен брой подобни по значение думи. Чрез използването на българския синонимен речник на slovored[5], ние забелязваме, че синонимите на “цел” са 30:

[същ.] намерение, мисъл, умисъл, план, стремление, стремеж, прицел, обект, резултат, край

[същ.] предназначение, назначение, определение

[същ.] идеал, мечта

[същ.] задача, интерес, право

[същ.] направление, насока, тенденция, смисъл

[същ.] мишена, висота

[същ.] причина, подбуда

[същ.] същност, същина, поанта

Този прост експеримент доказва, че един от големите недостатъци на алгоритъма е това, че контекстът не се взима предвид. Това е така, защото в този случай избирането на подходяща заместваща дума би се осъществило на случаен принцип. Липсата на отчитане на контекст показва, че проблемът не може да бъде решен с този подход.

2.2.1.2 *N-грами*

2.2.1.2.1 Описание на алгоритъма

N-грам [11] се нарича последователността от думи, които са N на брой. Пример за това са 2-грамите: “ходя бързо”, “голямо сърце”, “любовта е ”. Друго название за 2-грамите е биграми. Представители на 3-грамите са “ходя бързо напред”,
‘имаш голямо сърце” и други. Съответно прогресивно се увеличава бройката и така може да се разпишат и 4, 5 и 6-грами. Те могат да бъдат използвани за вземане

[5] www.slovored.com.https://slovored.com/search/synonymous/%D1%86%D0%B5%D0%BB

предвид на контекста преди думата, която търсим. По този начин лесно може да се реши проблемът за намиране на контекстов синоним на дадена чуждица. Чрез методи като вероятностен подход и Допускане на Марков [12] може да се използва силана на N-грамие за решаване на нужната задача.

Съществува и така нареченият Български национален корпус [13], който съдържа данни в себе си за околок един милиард думи, които са преобразумани в 1, 2, 3, 4 и 5-грами. Обучаването е направено спрямо думи в основна форма, както и думи в различни форми, което спомага за широкото използване на този модел на български.

Java for N-gram Generation

This code block generates n-grams at a sentence level. The input consists of **N** (the size of n-gram), **sent** the sentence and **ngramList** a place to store the n-grams generated.

```
private static void generateNgrams(int N, String sent, List ngramList)
  String[] tokens = sent.split("\\s+"); //split sentence into tokens

  //GENERATE THE N-GRAMS
  for(int k=0; k<(tokens.length-N+1); k++){
    String s="";
    int start=k;
    int end=k+N;
    for(int j=start; j<end; j++){
     s=s+""+tokens[j];
    }
    //Add n-gram to a list
    ngramList.add(s);
  }
}//End of method
```

2.2.1.2.2 Предимства и недостатъци на алгоритъма

Основаният на N-грами алгоитъм за намиране на липсващата част в дадено изречение има няколко плюса. Когато предвид се взима малко N – 2, 3 или 4 и N-грамите са предварително изчислени, то тогава чрез използването на вероятностният подход бързината на обработка би се вдигнала в пъти. Скоростта е един от важните признаци при избиране на начин за решаване на дадена задача.

Недостатък на този тип подход е, че за реализацията му е нужно да има предварително изчислени N-гради, както и голямо наличие на информация, с която да се борави по време на работа. В този корпус трябва да се съдържат и рядки думи, тъй като хората, използващи разширението ще има нужда да сменят

точно тях в писанието си. Най-големият недостатък в случая се явява липсата на отчитане на контекст след целевата дума. По този начин това ще намали точността на алторитъма и той може да се окаже неефективен за решаването на текущата задача.

2.2.1.3 *Изграждане на езиков модел с помощта на вграждания на думи*

2.2.1.3.1 Описание на алгоритъма

Вграждане на дума (*word embedding)* [14] е термин, който се използва в обработката на естествен език за да се назове представянето на дума, текст или изречение по научен начин, по който думите, които имат сходно представяне, да имат и сходно значение. Тези части на речта обикновено се представят като вектори с реални координати, които се намират в линейно пространство. Всеки вектор се отнася към една дума, като думите се научават по подобие на тренирането на една невронна мрежа. Тези вграждания мога да бъдат извлечени като се използват различни видове научаване на характеристики или езикови модели. Размерността на самите вектори не е много голяма. Броят на всички думи в речника е значително по-голям от нея. Всяко измерение се явява определен белег на думата, имайки предвид, че тези белези не са определени явно.

В основата на този подход стои хипотезата за разпределението [15]. Тази теория гласи, че думите, използвани в близък или сходен смисъл, имат сходни значения. При трениране на алгоритъм за вграждане, който използва много текстове, се получават подобни вектори за думи, използвани в сходни контексти. Има много видове алгоритми, с които може да се работи в този случай, като например цитираният в статията [16], както и Continuous Bag of words и (CBOW) и Continuous Skip-gram Model [17]. Действията на двата алгоритъма Skip-gram и CBOW са противоположни, като първият има за цел да предскаже какъв е контекстът на дадена дума спрямо самата чуждица, а вторият по обратния начин. Докато моделът, описан в статияна на Джейсън Браунлий работи в посока обработване на голямо количество текст и генериране на нов такъв спрямо наученото до тук. По този начин алгоритъмът може да се настрои да работи в полза на задачата на Разширението за текстови редактори и да спомага за намирането на контекста на чуждата дума, която трябва да бъде заменена.

```
# define model
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=seq_length))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(100, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))
print(model.summary())
```

За вграждания на български език може да се използват моделите на fasttext [18] или Wiki Word Vectors [19].

2.2.1.3.2 Предимства и недостатъци на алгоритъма

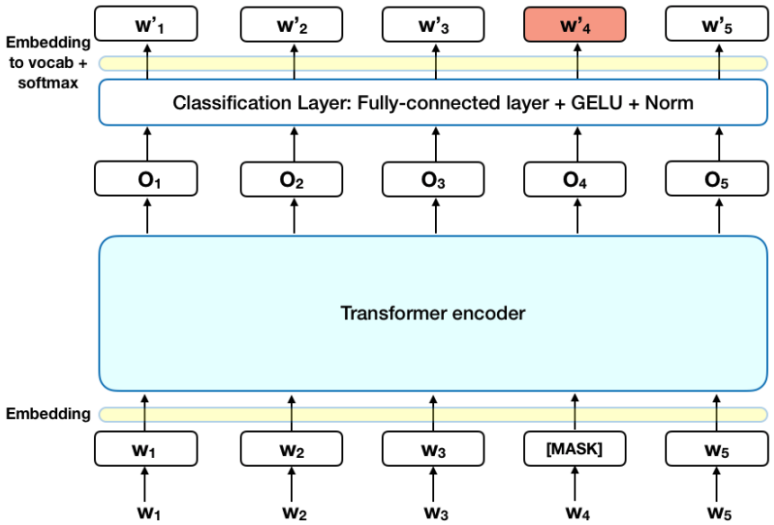
При алгоритмите, при които се работи с вграждания на думи основните недостатъци са два. Първият от тях е, че при тренирането с думи с голям брой значения подходът няма да прави разлика между тях и векторът по всяка вероятност ще е среден вариант на смисълът им. Бързодействието е вторият минус на този начин на разработка. Поради огромната нужда от данни при тренирането и научаването този алгоритъм изпитва сериозни трудности при развиването на добра скорост за предварителна подготовка на данните.

Предимство на алгоритъма е, че той взима предвид целият контекст на около целевата дума. Този алгоритъм е вероятностен и той ще бъде доста по-ефективен от другите, а доста лесно може да се нагоди към изискванията на задачата, която решаваме.

2.2.1.4 *BERT*

2.2.1.4.1 Описание на алгоритъма

Bidirectional Encoder Representations from Transformers (BERT) [20] е техника за машинно самобучение, която се използва като основен подход в обработката на естествен език и е тренирана предварително от Google. Създаден е през 2018 от Джъйкъп Девлин и негови колеги от Google. Архитектурата му [21] се описва като кодиращ трансформато, който е многослоен и двупосочен. Използва се така нареченият механизъм за внимание. Моделът е предразначен за решаването на две задачи – такава, която служи за предсказване на липсващата дума и в текст и тази, която предсказва следващото изречение. Първият вариант много наподобява нашия и лесно би се вписал в изискванията, които имаме.



2.2.1.4.2 Предимства и недостатъци

Предимствата на BERT[22] моделът са това, че той е изключително мощен и добре трениран от разработчиците си, като включва българския език и още 103 други. Информацията, използвана за тази цел се помещава в Уикипедия, но статиите на българския език към момента на трениране са били крайно недостатъчни за да бъде добре обучи добре алгоритъмът.

Описаният в предишното изречение минус играе важна роля в избирането на подход за разработка. Към него може да се добави и негативният ефект от бавното трениране и липсата на ресурс за самоинициативно обучение на BERT модел.

2.2.2 Възможни технологии, нужни за изграждането на разширението

Определянето метод за разработка на разширението за текстови редактори зависи до голяма степен от офисът, който ще бъде обслужван от рарзаботката. Като възможни продукти, ще разгледам пакетите от инструменти на OpenOffice, LibreOffice и Microsoft Office.

2.2.2.1 *OpenOffice*

Apache OpenOffice (AOO)[6] е система с отворен код, която се явява пакет от софуерни инструменти. Съдържа текстов редактор (Writer), електронни таблици (Calc), приложение, управляващо презентации (Impress), продукт служещ за рисуване (Draw), редактор за формули (Math) и система за управление на бази от данни (Base). Съвкупността от програми притежават собствени формати, серфицирани по ISO стандарти, но също така работят и с други, управлявани от Microsoft. OpenOffice е създаден, за да работи на различни платформи като например Linux, macOS and Windows, с възможностти за използване и от други.

Политиката за обновяване и разширяване на възможностите на OpenOffice е доста благоприятна и настроена позитивно към разработчиците, които искат да помагат.[7] За да създават новости не е нужно да се променя основния код, като разбира се всеки, който иска да промени източниците на базисно ниво е добре дошъл. Използването на UNO bridges дава възможност да се използват различни ресурси, за да се развива платформата. Някои от тях са Python, Java, Basic, C++, Visual FoxPro както и не програмистки разширения и много други благини, описани на сайта на продукта. По темата има изписани доста книги и статии дори цели работни рамки на съответните езици, които да спомогнат развитието на подобен научен труд.

[6] Apache OpenOffice. https://en.wikipedia.org/wiki/Apache_OpenOffice
[7] OpenOffice Wiki. https://wiki.openoffice.org/wiki/Extensions_development

2.2.2.2 *LibreOffice*

LibreOffice[8] е мощен инструмен на The Document Foundation (TDF), който се популяризира, като офис пакет, съпоставящ се на Microsoft Office. Тази съвкупност от софтуер се състои от програми за обработка на текст, електронни таблици, слайдшоута, диаграми, рисунки и дори бази от данни. Разпространена е на 115 езика, като нейните файлови формати са сертифицирани от ISO, по подобие на OpenOffice, с които доста си приличат по разработка, тъй като първоизточникът им е един и същ. LibreOffice също е мултиплатформен софтуер, който работи на разнообразни платформи като Microsoft Windows, MacOS, Linux, Android, iOS and Chromebook. Той също така има и онлайн версия, която се нарича LibreOffice Online, която включва апликациите Writer, Calc и Impress. Офис пакетът е най-популярният, който се използва в разпространените версии на Linux, като инсталационните файлове му са теглени над 7.5 милиона пъти за първите шест месеца от пускането на първата стабилна версия.

LibreOffice има 3 вида разширения:

- За разработчици на добавки, може да се използват няколко езика като например, Basic, Python, JavaScript, Java и BeanShell. За разработване на UNO допълнения може да се ползва и C++.
- Специални разширения Calc, известни като добавки. Те добавят функции към електронните таблици.
- Комплектовани разширения за шаблони и галерии.

Най-общо казано нужните ресурси за подобна разработка са описаните по-горе езици за програмиране, като в официалната документация доста детайлно е описано как може да се допринесе към развитието на пакета.

2.2.2.3 *Microsoft Office*

Всеизвестният на всички Microsoft Office е пакет от приложения с богато портфолио на функционалности. Някои от тях са Word, Excel, Access, PowePoint, Outlook и други. Групата е представена за пръв път през 1989 година, като е предназначена за Microsoft Windows, Windows Phone, Android, iOS macOS и Linux, което го прави мултиплатформен. Интерфейсът е достъпен на над 48 езика. Наличието на фунцкионалности и услуги, които се предлагат от софтуера се регулират спрямо лиценза, който е закупил потребителят. Office пакетът е широко разпространен комплект от апликации, който се развива с голяма скорост, за което допринася и общността от девелъпъри, които спомагат за

[8] LibreOffice. https://bg.libreoffice.org/

създаването на все повече и повече добавки към офис пакета. Нови версии се пускат през година, а всеки път вложеният ресурс става все по-голям. Microsoft Office също предлага и онлайн версия на своите продукти, като така документите могат да бъдат достъпвани отвсякъде с нужната интернет връзка. Големият плюс на тази платформа е връзката с училища и университети, която развиват и лицензите, които се отпускат на учещите се млади хора. [25]

За разработката на разширения[9] за Microsoft Office и в частност Word са нужни познания по HTML, CSS, and JavaScript, като с тези технологии лесно може да се създаде решение, което да е валидно онлайн, за Windows, за Mac и дори за iPad. Чрез тези основни технологии се дава възможност да се упражни build, test, debug, and publish върху така наречените Word add-ins.

2.2.2.4 Сравнение на технологиите

Microsoft Office години наред е била безспорно най-мощната платформа сред офис пакетите, но с нарастването на времето и на възможностите на технологиите на световния пазар се появиха и безплатни алтернативи като LibreOffice и OpenOffice. [26] Изборът между лицензиран и свободен софтуер зависи от нуждите на самия потребител. Разликите между LibreOffice и Apache OpenOffice са доста минимални, тъй като в основата си и двата инструмента произхождат от един и същи първоначален програмен код. Има няколко фактора, които могад да спомогна в разграничаването между трите платформи – характеристики, системни изисквания, съвместимост, цена и сигурност.

- Характеристиките и на трите продукта са сходни. Microsoft office има по-съвременен интерфейс с табове и туулбарове, докато другите офиси имат по-традиционен стил на окрасяване. Що се отнася до проверка на граматиката и правописа Microsoft имат вградени инструменти, докато при другите платформи трябва да се свалят допълнително. Libre Office и Open Office имат сходни функционалности със едни и същи имена.
- И трите пакета работят на повечето системи, като трябва да се има предвид, че LibreOffice и OpenOffice работят по-добре на Linux, отколкото другият вариант. Освен това, безплатните софтуери са за препоръчване, когато се ще се работи и с по-стари системи, които не могат да покрият нужните системни изисквания за платените разработки.
- Цената е една от основните разлики между трите. Libre Office и Open Office са безплатни, като за да притежава Microsoft Office даден

[9] Microsoft - Office Add-ins platform overview. https://docs.microsoft.com/en-us/office/dev/add-ins/overview/office-add-ins

потребител трябва да плати солидна сума. Това би наклонило везните в едната посока доста сериозно.

- Сигурността и в трите случая е на ниво, като трябва да се има предвид, че Microsoft Office държат кода си скрит, за да се предпазят от хакерски атаки.

Като заключение и трите офис пакета дават солидна основа за разработване на нужните функционалности. Един от основните фактори за избирането на един от трите продукта е цената, ако парите не са проблем – Microsoft Office, ако се иска нещо добро за цената си - Libre Office и Open Office. Друга основна причина може да е средата, която е устроена вече на машината, поради факта, че продуктите на Microsoft се интегрират много по-лесно едни с други, отколкото с програми на различни производители. Последното нещо, което оказва влияние са технологиите, използвани за разработване на разширенията. При трите платформи те са сходни, като само при Microsoft Office са HTML, CSS и JavaScript.

2.3 Езикови ресурси

2.3.1Речник на чуждите думи в българския език

Речникът на чуждите думи в българския език[10] е важен за системата, която разработваме, защото той може да бъде много полезен в различните аспекти на изграждане на системата. Той е основополагащ фактор за първия описан подход в точка 2.3.1.1 – *“Предлагане на заместващ синоним от синонимен речник ”*.

Важно негово приложение е, че думите от него могат да бъдат използвани с такава цел, че да се предотврати заместването на една чуждица с друга такава. Той също така може да бъде използван и за автоматичното засичане на чуждици в даден текст и по този тачин алгоритъмът да прихваща всички целеви думи и да предлага техни заместници.

Последна важна насока, в която можем да използваме речника на чуждите думи е, че той може да се съчетае с друг речник, който съдържа синоними на думи от българския език. С този подход може да се събере достатъчно голям корпус от заместващи думи, с които да борави системата.

2.3.2Стопдуми в българския език

Стопдумите [27] се описват като думите, които са най-често използвани в даден език. При Обработката на естествен език тези частици се филтрират непосредствено преди да започне процесът на работа на даден алгоритъм. Това е така, защото се счита, че тъй като тези думи са най-често срещани в даден език те допринасят най-малко за съответния контекст. Българският корпус от стопдуми [28] съдържа 259 думи, които могат да бъдат полезни при обработката на текстовете от разширението за текстови редактори.

2.3.3Синонимен речник

Наличието на синонимен речник ще спомогне за реализирането на целта на дипломната работа, като изиграе важна роля при правене на необходимите проучвания.

Синоними[10] са думи, които са близки по значение, но различни по състав спрямо други думи. Също така те имат еднакво или сходно значение в много контексти. От това определение може да си извадим като извод, че два синонима не е задължително да има еднакви значения в различни контексти. Проблемът с разпознаването на значението на дадената дума се пада на разширението за текстови редактори. Чрез ползването на този речник може да се изгради много бърз алгоритъм, който да спомага замяната на целевата дума. Този тип замяна е проста, но нейното бързодействие компенсира другите минуси. Начинът на използване е описан по-горе в точка две.

[10] Синоними - http://rechnik.info/%D1%81%D0%B8%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC

3 Използвани технологии

За целта на дипломната работа - разработването на разширение за текстови редактори, ще се използват различни софтуерни технологии, които ще бъдат описани в трета точка.

3.1 Софтуерни архитектури

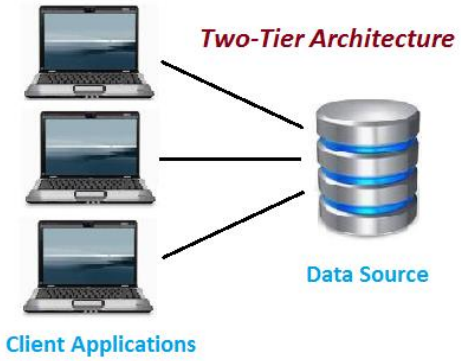
В разработката на приложения под софтуерната архитектура се разбира съвкупност от важни решения за организацията на програмните системи. Връзката между различните технологии и начинът, по който ще си взаимодействат те за да съществува една програма или приложение, комплектовано с документация се нарича архитектура на даден софтуер. Архитектурите са много видове, но ние ще разгледаме някои основни.

3.1.1Еднослойна архитектура

Еднослойната архитектура [29] се нарича тази, при която всички елементи на приложението (интерфейса, средния слой и обработката на данните) се намират на едно място или в един пакет. Този тип разработки се считат за най-прости за направа, но нуждата от нещо по-сложно се изпитва, когато става въпрос за създаване на уеб приложения или услуги, базирани на облачни системи. Съществуват доста варианти, в които простотата на разработката е важна, но сигурността, по-добрата скорост и по-лесното добавяне на функционалности са плюсовете на архитектурите с повече слоеве.

3.1.2Двуслойна архитектура

Двуслойната архитектура [30] се характеризира с това, че се състои от два слоя – презентационен слой (интерфейс), изпълняващ се на клиент и слой на данните, помещаващ се на сървър. Сигурността, производителността и бързодействието се уверичават с по-големия брой слоеве, но това коства и повече усилия и средства.



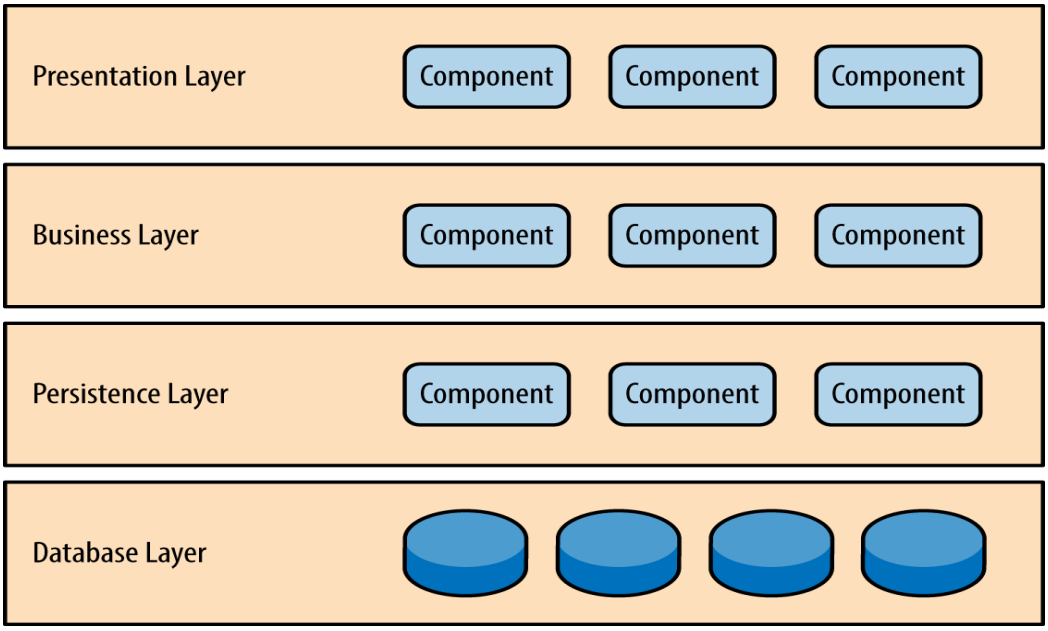
Фиг. 1 Двуслойна архитектура

3.1.3Многослойна архитектура

Многослойната архитектура [31] или още N-слойна архитектура е тази, при която приложението се разделя на три нива - презентационно ниво, логическо ниво и ниво на данни. Това е разделяне на различните части на приложението, за разлика от обикновено, концептуално или логическото разделяне на елементите при изграждане на на model-view-controller (MVC) архитектура. Друга разлика от MVC е, че слоевете от n-ниво са свързани линейно, което означава, че цялата комуникация трябва да мине през средния слой, който е логическият. В MVC няма така-наречен среден слой, защото взаимодействието е на три страни - контролният слой има достъп както до изгледа, така и до слоевете на модела и моделът също има достъп до изгледа, контролерът също така създава модел въз основа на изискванията и го подава към изгледа.

Предимствата на n-слойната архитектура включват:

- Мащабируемост
- Индивидуално управление, изолирана поддръжка
- Гъвкавост – промяна в зависивост от изискванията
- Сигурност – всеки слой има своя собствена защита.



Фиг. 2 Многослойна архитектура

3.2 База от данни

Базата данни[11] представлява колекция от данни. Обикновено данните са организирани така, че да представят модел от реалността. Тези данни могат да бъдат съхранявани на различни носители – хартия, електронни устройства, магнитни и оптични дискове и др.

Когато става въпрос за компютърни бази от данни се има предвид софтуер, който обслужва данните и предоставя достъп на външни приложения или потребители до тях. Този достъп може да бъде чрез потребителски интерфейс (за потребителите) или чрез стандартизирани интерфейси за програмен достъп – т.нар. драйвери от различни типове (ODBC, JDBC и т.н.).

Една система за управление на бази от данни (СУБД) има за цел да съхранява данни, позволявайки операции с тях – добавяне, изтриване, промяна и извличане. Широко известни към момента в световен мащаб системи са Oracle, IBM DB2, Sybase, Microsoft SQL Server, MySQL, MongoDB и много други.

Приложението на базите данни днес е на практика във всяка една област, в която се използва компютъризирано съхранение и обработка на данни – финанси, обучение, телекомуникации, магазини и т.н.

[11] Какво е база от данни? https://delc2.fmi.uni-plovdiv.net/courseCategory/show/8

3.3 Back-end технологии

3.3.1Python

Python[12] е един от най-популярните програмни езици, който се използва в много и различни сфери – от разработката на уеб приложения и такива за десктоп среда, до анализ на данни и управление на обучителни модели за машинно обучение (machine learning). Той е програмен език с общо предназначение, което допринася за широкото му разпространение в света на разработването на софтуер. Python е много по-лесен за научаване, сравнен с други програмни езици. Приложенията, написани на Python, са компактни и с лесно четим код за разработчиците, като често са и по-кратки от еквивалентните им, написани например на C/C++.

Python е известен с това, че е поставян на първо място сред разработчиците заради няколко неща:

- простотата на самия език
- че е лесен за научаване
- обществото от програмисти е достатъчно голямо
- най-вече големият избор от библиотеки с отворен код, които спомагат в сферите на уеб разработки, машинно самообучение, визуализация на данни.

Някои от тези библиотеки са: Scikit-learn, NumPy, Scrapy, TensorFlow, Pandas, Matplotlib, Seaborn, NLTK и cx_Oracle.

3.3.1.1 *Scikit-learn*

Scikit-learn[13] (по-рано scikits.learn и известен също като sklearn) е безплатна библиотека с отворен код за машинно обучение написана на езика за програмиране Python. Тя разполага с различни алгоритми за класификация, регресия и клъстериране, random forests, k-means. Също така тя е проектирана да си взаимодейства с числовите и научни библиотеки NumPy и SciPy.

[12] За какво се използва Python и какви са ползите му? https://blog.superhosting.bg/why-and-for-what-python.html

[13] Официална документация на Scikit-learn - https://scikit-learn.org/stable/getting_started.html

3.3.1.2 *NumPy*

NumPy[14] е основният пакет за научни изчисления в Python. Това е библиотека на Python, която осигурява многомерен масив, различни наследени обекти (като маскирани масиви и матрици) и асортимент от действия за бързи операции върху масиви, включително математически, логически, манипулиране на форми, сортиране, дискретни преобразувания, основна линейна алгебра, основни статистически операции, произволна симулация и много други.

В основата на пакета NumPy е обектът ndarray. Това капсулира n-мерни масиви от хомогенни типове данни, като много операции се изпълняват в компилиран код за изпълнение. Има няколко важни разлики между масивите в NumPy и стандартните типове данни, използвани в Python. Някои от тях са:

- Масивите на NumPy имат предварително указана дължина за разлика от листовете в Python. При промяната на големината на ndarray старата структура се трие и се създава нова.
- Елементите в NumPy масивите задължително трябва да са с еднаква дължина, което означава и че ще заемат еднакаква част от паметта.
- NumPy предлага голяма ефективост при извършване на математически и логически операции с големи масиви от данни.

3.3.1.3 *Scrapy*

Scrapy [15]е работна рамка за обхождане на уеб сайтове и извличането на данни в структуриран вид. Тази библиотека има приложение в обработването на данни или информация. Въпреки че, Scrapy е било създадено за извлиачне на информация от уеб сайтове, той може да бъде използван и за извличане на информация чрез API-та (например някой от продуктите на Амазон) или за обхождане на уеб страници.

```
import scrapy


class QuotesSpider(scrapy.Spider):
    name = 'quotes'
    start_urls = [
        'http://quotes.toscrape.com/tag/humor/',
    ]

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'author': quote.xpath('span/small/text()').get(),
                'text': quote.css('span.text::text').get(),
            }
```

[14] Официална документация на NumPy - https://numpy.org/doc/stable/user/whatisnumpy.html
[15] Официална документация на Scrapy - https://docs.scrapy.org/en/latest/index.html

```
        next_page = response.css('li.next a::attr("href")').get()
        if next_page is not None:
            yield response.follow(next_page, self.parse)
```

scrapy runspider quotes_spider.py -o quotes.jl

Фиг. 3 разписване на класа и извикване на скрипта

При извикването на командата scrapy runspider quotes_spider.py -o quotes.jl, в системата се проверява за дефиниция на така наречения *паяк*, като ако намери такъв се изпълнява и самият crawler (събирачът на данни от уеб сайтове). В start_urls атрибута се задават стойности за страниците, от които ще се извлича инхормация, в parse метода свалените елементи се завъртат в цикъл, използвайки така наречените CSS селектори, с които се идентифицират нужните обекти за обработка. Заявките към URL-а стават асинхронно, което спомага няколко неща да се правят паралелно и вдига бързодействието. Някои важни характеристики на работната рамка са:

- Чрез CSS селектори и XPath изрази се поддържа извличането на данни от HTML или XML източници
- Лесна за работа конзола IPython aware, с която удобно се пробват изразите
- Вградена поддръжка за създаването на извадки в различни формати (JSON, CSV, XML)
- Telnet конзола, която лесно се свързва тази на Python с процеса на Scrapy.

3.3.1.4 *TensorFlow*

TensorFlow [16] е библиотека с отворен код за бързи цифрови изчисления. Тя е създадена и се поддържа от Google и е издадена под лицензa Apache 2.0 с отворен код. API-то е разработено за езика за програмиране Python, въпреки че има достъп до основния API на C ++.

За разлика от други цифрови библиотеки, предназначени за използване в Deep Learning като Theano, TensorFlow е проектиран за използване както в научноизследователска и развойна дейност, така и в производствени системи, не на последно място RankBrain в търсенето с Google. Може да работи на единични CPU системи, графични процесори, както и мобилни устройства и широкомащабни разпределени системи на голям брой машини.

[16] Introduction to the Python Deep Learning Library TensorFlow - https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/

Чрез TensorFlow могат да се създават големи невронни мрежи, които имат много на брой слоеве. Поради тази причина той е един от основните инструменти в областта на дълбокото обучение с Python. Библиотеката се използва най-често за регресия, класификация, създаване и предсказване.

3.3.1.5 *Pandas*

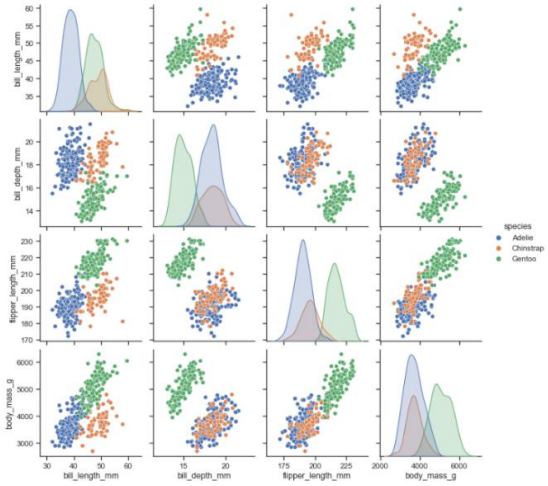
Софтуерната библиотека Pandas [17] е написана на Python за манипулиране и анализ на данни. Тя е с отворен код, което позволява свободното й разпространие. Основни нейни характеристики са:

- *DataFrame* обект за манипулиране на данни с интегрирано индексиране
- Инструменти за четене и запис на данни между структури от паметта и различни файлови формати
- Интегрирана обработка на липсващи данни.
- Преобразуване на набори от данни
- Индексиране на големи набори от данни
- Вмъкване и изтриване на колони от данни
- Групирайте по област, позволявайки операции за разделяне, прилагане и комбиниране на набори от данни
- Сливане и групиране на набор от данни
- Функционалност за работа с време: Генериране на диапазон от дати, преместване на дати и изоставане спрямо времевите зони
- Осигурява филтриране на данни

3.3.1.6 *Seaborn*

Един от най-известните начини за визуалицазия на данни е библиотеката на Python – Seaborn[18]. Тази софтуерна разработка служи за изготвянето на статистически графики. Тя е изградена на базата на Matplotlib и е интегрирана с структурите от данни на Pandas. Тя включва в себе си различни функционалности като API, което служи за разглеждане на връзки между различни променливи. Също така в нея се съдържа и поддръжка за променливи, които са разделени на категории с цел наблюдение или събиране на статистики, както и за сравняването на различни множества от данни. Други важни характеристики са удобни изгледи за сравняване на сложи структури от данни в различни форми и цветове. На фиг. 4 са показани различните видове графики, разработени чрез инструментите на Seaborn.

[17] Официална документация на Pandas - https://pandas.pydata.org/about/index.html
[18] Документация за Seaborn - https://seaborn.pydata.org/



Фиг. 4 различни видове графики на Seaborn

3.3.1.7 *NLTK*

Една от водещите платформи по създаване на приложения на Python, които да работят с човешки език се казва *NLTK - Natural Language Toolkit*[19]. Тя предоставя лесни за работа интерфейси, съдържащи безброй много лексикални ресурси като WordNet. Тези мощни инструменти слущат за обработка на текст, класификация, токенизация, stemming, tagging, parsing. Библиотеката работи добре и със семантични операции. Тя е налична за Windows, Mac OS X и Linux, което доказва гъвкавото ѝ приложение.

3.3.1.8 *Cx_Oracle*

Cx_Oracle[20] е модул, който позволява достъп до база от данни на Oracle и отговаря на спецификацията на API за Python. Понастоящем този модул е тестван за разработки на Oracle Client 21c, 19c, 18c, 12c и 11.2 и Python 3.6, 3.7, 3.8 и 3.9. По-стари версии на cx_Oracle могат да се използват с предишни версии на Python. cx_Oracle се разпространява под лиценз с отворен код (BSD лиценз). Модулът бива извикан от Python скриптовете, като вътрешно той зарежда динамично Oracle Client библиотеки, които достъпват базата от данни. Тя може да е на същата машина или да се намира на определен сървър. Cx_Oracle се инсталира през pip, но самите библиотеки на Oracle трябва да се инсталират отделно. Разработката спомага лесно да се реализират CRUD (create, retrieve, update и delete) операциите, което прави приложението по-динамично и лесно разшируемо.

[19] Официална документация на NLTK - https://www.nltk.org/
[20] Сайт на cx_Oracle - https://cx-oracle.readthedocs.io/en/latest/

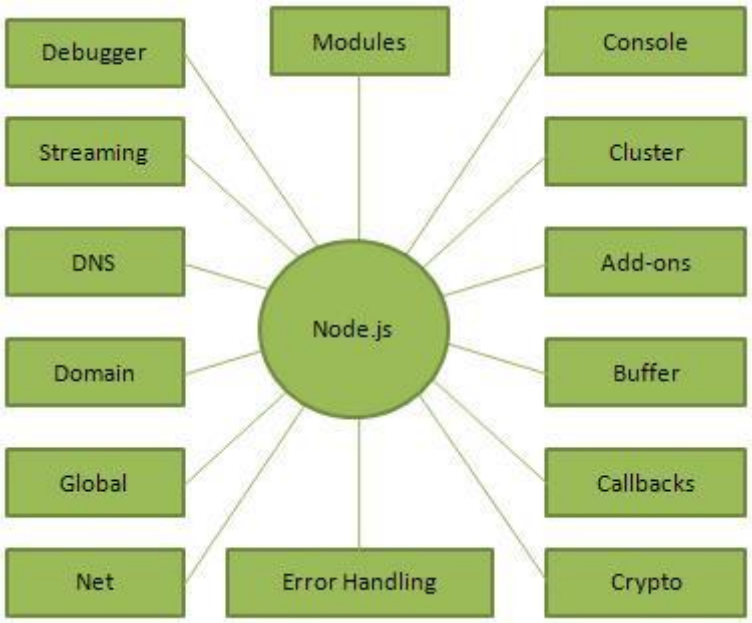
3.3.1.9 *Node.js*

Node.js [21]е сървърно базирана платформа, изградена на JavaScript(V8). Тя е система с отворен код, която работи на различни операционни системи. Средата за разработване служи за създаване както на така-наречените server-side приложения, така и за мрежови такива. Основен инструмен за работа е JavaScript, като добавка към характеристиките на системата се взимат предвид и някои модули, писани на езика, които улесняват разработката на уеб приложения. Някои от основните плюсове на работната рамка са:

- Асинхронна разработка със събитийно програмиране - всички API-та на библиотеката Node.js са асинхронни. По същество това означава, че сървър, базиран на Node.js, никога не чака дадено API да върне данни. Сървърът преминава към следващия API, след като го извика и механизмът за известяване за събития помага на сървъра да получи отговор от предишното извикване на API.
- Скоростта на действие на Node.js се дължи на основана, на която е написана – JavaScript(V8 Engine).
- Удобства при работа с RESTful заявки.
- Моделът на Node.js е еднонишков, но лесно скалируем. Механизмът за управление на събития спомага за по-бързото и лесно справяне със заявките към сървъра.
- Лицензът е на MIT
- Не се създава буфер на данните, а те се изкарват на парчета.

Моделът на работа на Node.js е много прост, но разделен на доста части, описани на фиг. 5

[21] Официална документация на Node.js - https://nodejs.org/en/docs/



Фиг. 5 модули на Node.js

3.4 Front-end технологии

Разработката на изгледът на разрширението за тектови редактори може да стане с някои от работните рамки, написани на JavaScript, които ще бъдат описани в тази точка.

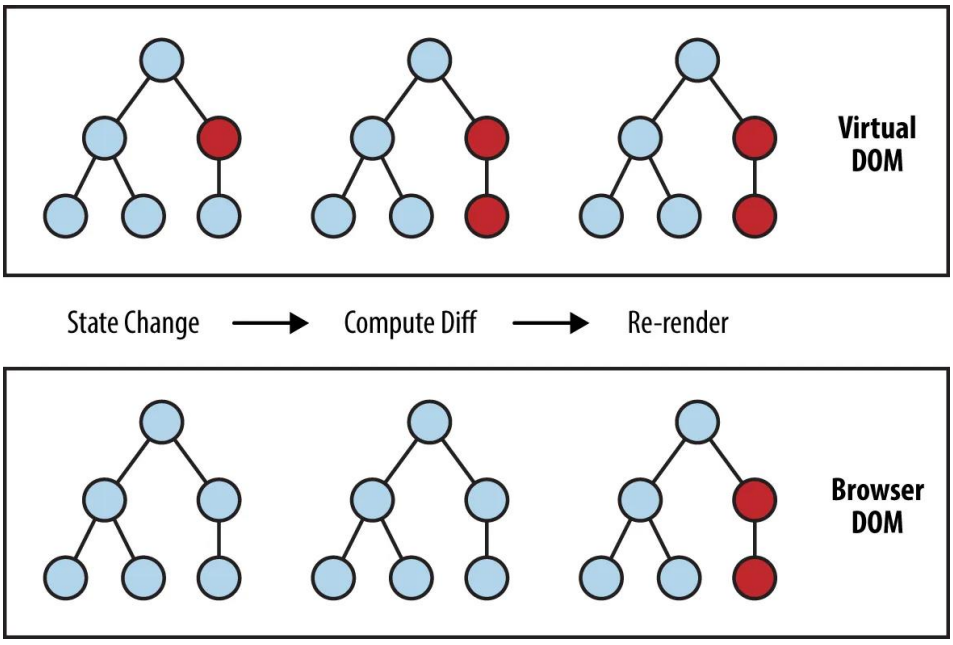
3.4.1ReactJS

ReactJS [22] е JavaScript библиотека за изграждане на потребителски интерфейс, която е с отворен код. За нейната поддръжка се грижат общност от компании и разработчици, както и хората от Facebook. Той може да бъде използван като основа при разработката на мобилни приложения или на уеб приложения, състоящи се от една страница. **React** е отговоен основно за уплавлението и доставянето на данни в **DOM** дървото, поради което създаването на приложения с него изисква и използването на допълнителни настройки и библиотеки, с които се обработва състоянието на елементите на графичната среда и елементите за маршутиризане между различните позиции в приложението.

Архитектурата на библиотеката е базирана на компоненти. Те се разделят на два вида – функционални и такива, базирани на класове. Първият вид се декларира чрез функция, която връща JSX, докато вторият чрез ECMAScript класове.

[22] Официална документация на React.js - https://reactjs.org/docs/getting-started.html

Друга важна характеристика на **ReactJS** е наличието на виртуален документен обектен модел или накратко виртуален DOM (virtual DOM). Библиотеката създава кеш за структура от данни в паметта, след което се правят нужните промени виртуално и се отразяват в браузъра. Този процес се нарича **reconciliation**. Това създава впечатление, че страницата се пресъздава всеки път, когато се прави промяна, а в действителност разликата се появява в компонентине на React. Това селективно изобразяване осигурява значително подобрение на производителността. Тази характеристика спестява усилията за преобразуване на CSS, промените по оформлението на страницата и изобразяването за цялата страница.



Фиг. 6 изчисленията, правени от виртуалния DOM

На фиг. 6 е изобразено как се правят преобразуванията от виртуалния DOM.

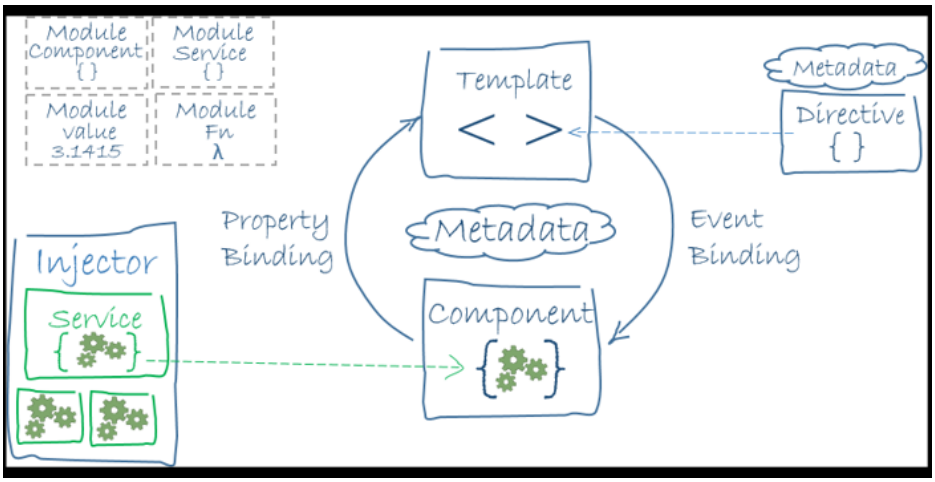
3.4.2Angular

Angular [23](по-рано наричан AngularJS) е работна рамка за разработка на уеб приложения, която е базирана на TypeScript. Тя е пренаписаната версия на AngularJS от група от хора в гугъл и общност от разработчици. Разликите между Angular и AngularJS се изразяват в няколко аспекта:

- Angular няма концепция за "scope" или контролери, вместо това използва йерархия на компонентите като основна архитектурна характеристика.

[23] Официална документация на Angular - https://angular.io/

- Angular има различен синтаксис на доста от изразите, като основният фокус се измества върху "[]" за свързване на свойства и "()" за свързване на събития.
- Модуларност – функционалността се разделя на модули.
- Angular препоръчва използването на TypeScript, което включва статично типизиране и анотиране.
- Динамично зареждане.
- Поддръжка на асинхронни операции.



Фиг. 7 Архитектура на Angular

Архитектурата на ***Angular*** лежи на няколко фундаментални концепции. Основните изграждащи единици на работната рамка са компонентите, които са комплектовани в NgModules. NgModules събират парчета код, които са свързани с цел образуване на функционални множества. Всяко приложение има поне един основен модул, който се използва, за да стартира bootstrap и много други модули, за да обособи важни функционалности. Компонентите определят изгледите (*views*), които са съвкупност от графични елементи, обработвани и подготвяни от Angular, за да представят данните, предоставени от по-долните слоеве. Компонентите използва и така наречените сървиси (*services*), предоставящи специфични функционалности, индиректно свързани с изгледите. Те често могат да бъдат добавени като зависимости, правейки кода преизползваем и ефективен.

Модулите, компонентите и сървисите са класове, наречени декоратори, предоставящи метаданни, които указва на Angular как да работи с функционалните единици. На фиг. 7 е описана архитектурата на самата работна рамка. Компонентите и шаблоните дефинират изгледите в Angular, като се предоставя и възможността за d*ependency injection* [32].

4 Анализ на изискванията

4.1 Концептуален модел

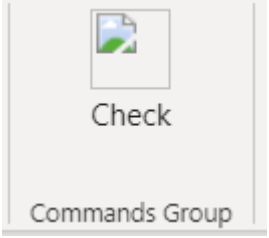
След обстойния анализ на областта във втора точка и описание на използваните технологии в трета се оформи един процес на работа, който може да се имплементира лесно в Разширението за текстови редактори. Базирайки се на методите и подходите, се цели да се разработи цялостно приложение, което да играе ролята на интерфейс, към който бързо и просто могат да се извикват различни методи за предлагане на чуждици, описани чрез алгоритмите за обработка на естествен език или други подобни.

Описаните начини за реализация могат да бъдат комбинирани по различни начини, за да се получи максимално ефективна система. В този случай, ще се използва подходът, в който за изглед ще се използва **Angular framework**, за сървърната част – **Node.js,** за база от данни – **Oracle PL/SQL** и за слой за обработка на данни **Python** и негови библиотеки.

4.2 Функционални изисквания

Вече се запознахме със функционалностите, които предлагат съществуващите подобни разработки. Те са част от първоначалния анализ проведен, за да може да се конструира системата. В тази подточка ще се спрем на функционалните изисквания, които трябва да има един **add-in**. Основните задачи поставени пред системата са:

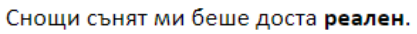
- Първоначален екран на приложението, от който се стартира проверката – Първоначалното състояние е доста просто. То се изразява в един бутон, който е направен така, че да прихваща текста в документа и да го подава към долните слоеве.



Фиг. 8 Бутон за стартиране

- Модул за свързване на изгледа със сървърната част – този модул е реализиран чрез Node.js и Angular. Той е използван като медиатор и не съдържа в себе си никаква бизнес логика.

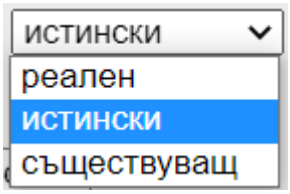
- Улеснение на потребителя да избере дали сам да стартира процеса при пускане на приложението или това да става чрез бутон “*Process*”.



Фиг. 9 Изход от системата

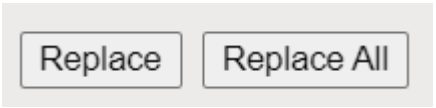
Фиг. 10 Контроли за работа

- Модул за комуникация на сървърната част с логическия слой.
- Модул за връзка между логическият слой и така нареченият data layer(слой, занимаващ се с данни).
- Обработка на подаденият текст и предлагане на заместваща дума за целевата
- Маркиране на чуждицата с удебелен шрифт, за да се отличава от другите думи.
- Изреждане на предложенията в тип списък (drop-down list).



Фиг. 11 Drop-down със синоними

- Предоставяне на възможност за заменяне на чуждицата на първото й появяване в текста със синоним или заменяне на всичките места, на които тя се намира в текста.



Фиг. 12 Бутони за заместване

- Модул за обработка на снимките на речникът на чуждите думи и създаване на електронен негов вариант.

Сценариите за упротреба са три – използване на системата за:

1. Показване на чуждиците в даден текст.
2. Предлагане и заместване на първото появяване на чужда дума.
3. Заместване на всички места дадената чужда дума с нейн синоним.

4.3 Нефункционални изисквания

4.3.1Модулност на системата

Проектът трябва да бъди разработен спрямо основните “постулати” и принципи на софтуерното инженерство. Разширението трябва да лесно да предоставя възможността към него да се закачат други подходи с цел подобряване на методите му на работа. Разделянето на разработката на модули дава възможност тя да бъде променяна без големи усилия, като да се добавят нови функционалности за малко време. Изгледите и потребителските екрани са изградени на същия принцип, като тяхната структура спазва моделите на разработка на цялото приложение.

4.3.2Съвместимост

Обстоятелствата към момента на разработка предполагат, че системата ще бъде съвместима с различни системи, поддържащи десктоп версията на Microsoft Word и инструментите за разработката на разширението. Допълнителна положителна страна тук е, че онлайн приложението на Microsoft Word работи на всякакви платформи и е ефективно решение за *Winows, Linux* и *MacOS*.

4.3.3Приложимост

Разширението е приложимо за всякакви групи от потребители, използващи текстори редактори за четене и писане на текст. С удобния си интерфейс той е лесен за употреба инструмент, използващ различни методи за откриване на синоними. Този модул към Microsoft Word намира приложение в различни сфери на обработка на текстове, например – при писане на статии.

4.3.4Разширяемост

Тук се има предвид бъдещото развитие на модула. Това е част от принципите на дизайна на софтуерни приложения. За по-нататъчно разработване може да се вземе предвид не само разширяване на текущите функции и добавянето на нови такива, но и евентуалното имплементиране на добавката към друг тип офиси – *LibreOffice, OpenOffice* като по този начин да се разшири неговото приложение.

5 Проектиране на приложението

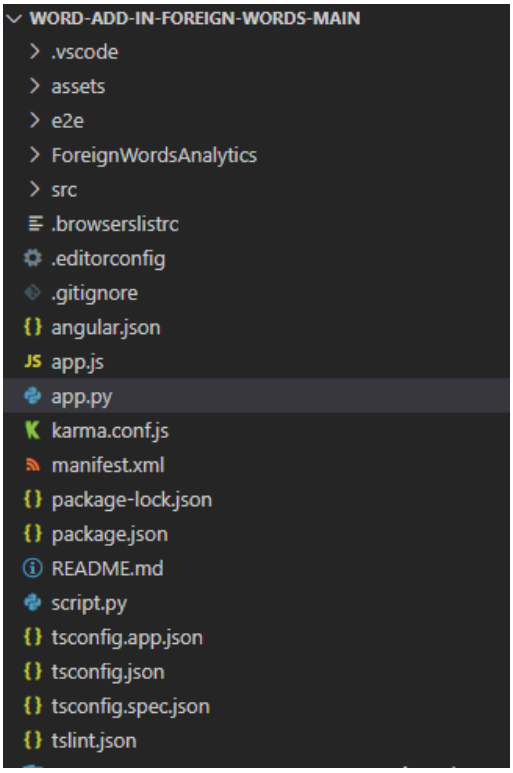
5.1 Избор на архитектура

Архитектурата на разширението за тектови редактори ще е многослойна. Тя ще се състои от слой на данните, в който ще бъдат подготвена и обработена нужната информация, логически слой, грижещ се за извършването на нужните операции за избирането на най-подходящ синоним, сървърен слой, свързващ клиентския слой с логическия слой и изглед, който ще представлява група от графични елементи, създадени за удобство при използване на разширението. Слоевете ще са подобни на тези, показани на фиг. 2.

За база от данни ще се използва *Oracle Express Edition 19c*, защото е безплатна и удобна за учебни цели. На *Python* ще се прави обработката на данните и предлагане не чуждица, тъй като към него има голям набор от мощни библиотеки и инструменти, подходящи за задачата , а на *Node.js* ще се разработва сървърното решение. Текстовият редактор, за който ще се създава разширение е *Microsoft Word*, защото той е най-разпространен и по този начин приложението ще може да се тества най-лесно. Изграждането ще се прави с помощта на *HTML, CSS, JavaScript* и посоченият по-горе *Angular framework*. За целта на по-лесни и бързи експерименти ще се използва методът описан в точка 2.3.1.1 - *Предлагане на заместващ синоним от речник*.

5.2 Файлова структура

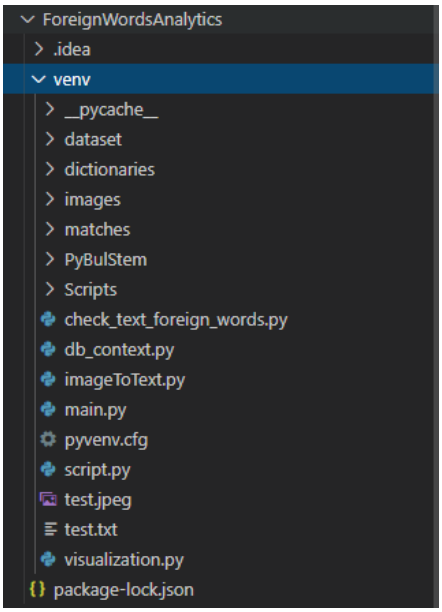
Файловата структура в проекта е организирана в няколко подпапки, разделени спрямо езиците, най които са написани. Програмният код е качен в GitHub репозитори, заедно с документацията, с цел съхранението му да става по-лесно и сигурността около запазването му да е по-голяма.



Фиг. 13 Основна структура на папката word-add-in-foreign-words-main

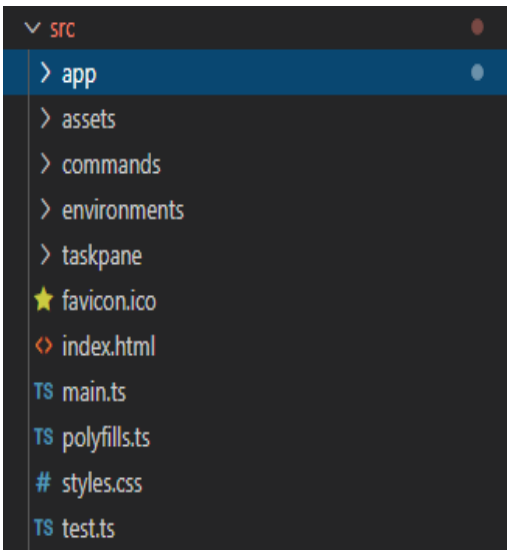
На фиг. 13 е изобразена основната структура на папката word-add-in-foreign-words-main.

В нея се виждат няколко важни файла като *app.js, app.py, script.py* и *manifest.xml*. *App.py* и *script.py* са разписани с тестови цели, за да се пробва връзката между различните компоненти на разработката. *App.js* е основният файл на *Node.js,* на който са описани структурите и командите, използвани за стартиране и управление на сървъра, а в *manifest.xml* чрез така наречения маркиращ език са описани функционалните единици на разширението, с цел то да се предава лесно през различните платформи. Всички файлове с разширение .json се явяват конфигурационни за разработката, като настройките са за различните нива и слоеве, които имат нужда от допълнителни инструкции. *Angular.json* има за цел да зададе основни стартиращи насоки за работната рамка *Angular,* като те се изпълняват при стартиране на приложението, а някои и при самото изпълнение на функционалностите.



На фиг. 14 са изобразени файловете в папката ForeignWordsAnalytics, които са написани на езика Python и държат в себе си основната логика за предлагане на синоним на целевата дума. Те ще бъдат описани по-подробно в следващите подточки.

Фиг. 14 Структурата на папката ForeignWordsAnalytics



На фиг. 15 виждаме структурана на папката src, в която има файлове, специфични за addin-а, като в тях има различни команди на HTML, CSS и JavaScript, които ще бъдат описани в следващите подточки.

Фиг. 15 Структура на папката src

5.3 Слой грижещ се за данните

Това ниво на архитектура се разделя на две части – едната е текстовите файлове, в които се съдържат думите от речниците, а другата е алтернативният вариант, характеризиращ се с това, че синонимите се съхраняват в база от данни.

5.3.1Текстови файлове

Текстовите файлове в проека играят важна роля при запазването на информацията от *Речник на чуждите думи в българския език* [10], тъй като в интернет не се намира лесно негов електронен вариант. За целта на разработката са използвани снимки на речника, от които се извличат нужните данни. Поради тази причина за голяма част от разработката на този слой от приложението е използван продуктът на Google Tesseract OCR [33]. Този софтуер, разработен от технологичния гигант и лицензиран под Apache License 2.0, е създаден с цел да се извлича инфорамация от снимки. Тази негова функционалност се интегрира лесно в Python, както е и показано в приложението във файла /ForeignWordsAnalytics/venv/imageToText.py в метода extract_data_from_image(). В него е описана простата команда от библиотеката pytesseract - image_to_string, на която за първи параметър се подава пътят до снимката, от която искаме да се извлече информацията и езикът, на който е написана думата и обясненията към нея. Една част от снимките имаха нужда от така-нареченият pre-processing (предварителна обработка) преди да бъдат подадени на метода за извличане на текст.

```
def extract_data_from_image():
    # simple version for working with CWD
    DIR = 'D:/PythonProjects/venv/images/cut'
    num_of_files = len([name for name in os.listdir(DIR) if os.path.isfile(os.
path.join(DIR, name))])
    pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
OCR\tesseract'

    for i in range(num_of_files):

        text = pytesseract.image_to_string(image = r'images/cut/' + str(i+1) +
 '.jpg', lang='bul')
        splitted = text.splitlines()

        word = ''
        words = []

        previous = next_ = None
        l = len(splitted)
        for index, obj in enumerate(splitted):
            if splitted[index].endswith('-'):
                word = word + splitted[index][ : -1]
            else:
                word = word + splitted[index]
                if index > 0:
```

```
                previous = splitted[index - 1]
            if splitted[index] == '' and previous.endswith('.'):
               words.append(word)
               word = ''

    words = ["-".join(line.split(" ", 1)) for line in words]
    # print(*words, sep = '\n')
    with open('dictionaries/' + str(i+1) + '.txt', 'w') as f:
        for item in words:
            k = item.rfind(".")
            item = item.replace(';', ',')

            lang_field = prepare_item(item)

            start = item.find('(')
            end = item.find(')')
            if start != -1 and end != -1:
                item = item[:start] + item[end + 2:]

            f.write("%s\n" % (item[:k] + lang_field + ";"))
```

Фиг. 16 методът extract_data_from_image()

На фиг. 16 е показана функцията, която обхожда ред по ред извлечения текст и го обработва, като накрая се създава специално форматиран речник - ForeignWordsAnalytics/venv/dictionaries/formatted/fastCheckWordsFinal.txt.

Тук данните се пазят във подобен вид:

абаут-относно,за,отнасящ се за.#Произход:незнаен#Област:неидентифицирана#;

абориген-местен жител, туземец.#Произход:незнаен#Област:неидентифицирана#;

абсорбира-попива.#Произход:незнаен#Област:неидентифицирана#;

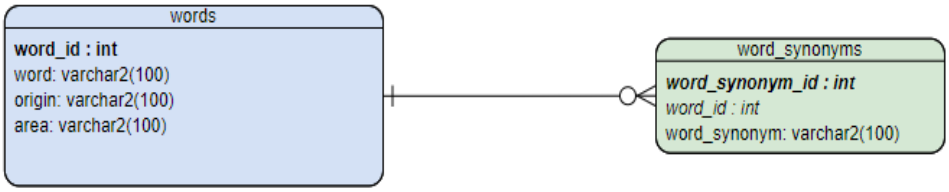
аванта-облага, придобита без влагане на труд и средства.#Произход:незнаен#Област:неидентифицирана#;

като разделението със знак # спомага да се добавят произход и област на дадената чуждица.

Във файлът fastCheckWordsFinal.txt се съдържат и думи, извлечени от сайтът www.чуждица.бг, като за целта е използвана библиотеката на Python – Scrapy версия 1.8, описана в точка 3.2.3.1.3. Чрез метод за събиране на HTML-а от уеб страници, разписан в библиотеката се отделят данните, които впоследствие се добавят към корпуса от думи след нужната предварителна обработка, извършена в кода на Python.

5.3.2База от данни

Както беше описано в точка 4.1 използваната база от данни в проекта е Oracle Express Edition 19c, тъй като тя е леснодостъпна и безплатна за учебни цели. Системата за управление на бази от данни се нарича PL/SQL developer – 14 тестова версия, като тя съдържа в себе си всички необходими функционалности за поддържане на нужните данни. Алгоритъмът от точка 4.3.2 се повтаря, като различното в случая е, че извлечената информация се записва в няколко таблици в базата, описани по-долу на фиг. 17.



Фиг. 17 Entity-relationship diagram

В таблица ***words*** има 4 колони – *word_id,* която е от целочислен тип и е първичен ключ, *word,* съдържаща в себе си думата, origin – *произходът* и *area* – областта всички от тип стринг. Втората таблица ***word_synonyms*** има първичен ключ - *word_synonym_id* от целочислен тип, външен ключ, който дефинира релация едно към много от таблицата ***words*** към ***word_synonyms*** чрез колоната *word_id* и текстовото поле *word_synonym*, съдържаща синонима на съответната чуждица.

```
import cx_Oracle

connection = None
try:
    connection = cx_Oracle.connect(
        "SYS",
        "password",
        "localhost:1522/XE",
        encoding="UTF-8",
        mode=cx_Oracle.SYSDBA)

    # show the version of the Oracle Database
    print(connection.version)
```

```
except cx_Oracle.Error as error:
    print(error)
finally:
    # release the connection
    if connection:
        connection.close()
```

Фиг. 18 db_context.py

Във файлът, описан на фиг. 18 е демонстрирано как се установява връзката между Python и базата от данни на Oracle, инсталирана на локална машина, иначе казано на localhost. За експериментът е използван encoding – UTF-8, потребителят SYS, който е със роля на администратор на базата от данни и портът – 1522, на който се реализира заявката. Използването на SYS потребителят за комуникация между слоевете не е добра практика, но в средата на тестови проект не е толкова голям проблем.

5.4 Логически слой

Рарзаботката на логическият слой се реализира изцяло в кода на Python, позициониран в папката ForeignWordsAnalytics. Основната задача на тази част на проекта е да получи като вход текстът, който трябва да бъде проверен и да върне като изход чуждиците, техните синоними , областта и произхода (ако са налични).

```
import check_text_foreign_words as check
import os
import sys

def process_text_dict(text):
    words_res = check.get_words_from_dict(text)
    return words_res

def main():
    res = process_text_dict(sys.argv[1])
    text = [ord(c) for c in res]
    print(text)

main()
```

Фиг. 19 main.py

Задачата се стартира от main метода във файла *main.py* (фиг. 19), който приема текстът, подаден като аргумент от по-горния слой и го подава на алгоритъма за проверка. В *check_text_foreign_words.py* се извиква функцията

fast_check, проверяваща за съдържащи се чужди думи в текста, като алгоритъмът е много прост с цел тестовите сценарии да се изпълняват по-бързо и лесно:

1. Текстът се разделя на отделни тоукъни чрез TweetTokenizer от библиотеката nltk (описано в точка 3.2.3.1.7).
2. Премахват се излишните стоп думи, чрез корпусът на български език [28].
3. Чрез PyBulStem[24] (интерпретация на BulStem[25] на Преслав Наков, пренаписан на друг език, за нуждите на приложението) отделните тоукъни минават през обработката, наречена stemming, привеждаща ги в основна форма, което спомага за по-точното прихващане на чуждиците.
4. На принципа pattern-matching се отсяват чуждите думи и се представят във вид – "чуждица – синоними – област - произход".
5. Генерира нужната информация и се подава обратно на горния слой.

5.5 Сървърна част

Основната работа на това ниво е да предаде информацията от изгледът към логическия слой. Тя е разписана чрез *Node.js*, като основната й част се намира във *app.js*, където са разписани няколко тестови заявки, за които да слуша сървърът и една реална GET заявка – (*/api/process:text*) , обръщаща се към Python *main.py* метода и подаваща му нужните данни.

```
const express = require('express');
const bodyParser = require('body-parser');
const {
    spawn
} = require('child_process');
var utf8 = require('utf8');
const path = require('path');
var fs = require('fs');
var http = require('http');
var https = require('https');
var privateKey = fs.readFileSync('node_modules/browser-
sync/certs/server.key', 'utf8');
var certificate = fs.readFileSync('node_modules/browser-
sync/certs/server.crt', 'utf8');
var credentials = {
    key: privateKey,
```

[24] PyBulStem - https://github.com/peio/PyBulStem/blob/master/bulstem.py
[25] Bulstem - http://lml.bas.bg/~nakov/bulstem/и

```
    cert: certificate
};

const app = express();
```

Фиг. 20 Променливи и константи в app.js

На фиг. 20 е показана декларацията на нужните за обработката на заявки към сървара променливи и константи. *Express*[26] е библиотека на JavaScript, която се използва за стартиране и конфигуриране на http и https сървър. На privateKey, certificate и certificate са присвоени нужните метаданни, за да се стартира успешно приложението, “слушащо” за заявки към него на порт 4200, написано в app.user метода. В тази част на кода се задават и други настройки нужни за обработката на заявките – фиг. 21.

```
// Website you wish to allow to connect
    res.setHeader('Access-Control-Allow-Origin', 'https://localhost:4200');

    // Request methods you wish to allow
    res.setHeader('Access-Control-Allow-
Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');

    // Request headers you wish to allow
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-
type');

    // Set to true if you need the website to include cookies in the requests
sent
    // to the API (e.g. in case you use sessions)
    res.setHeader('Access-Control-Allow-Credentials', true);

var httpsServer = https.createServer(credentials, app);

httpsServer.listen(3000);
```

Фиг. 21 настройки на HTTPS сървъра

Стартирането става чрез две команди – createServer и listen, описани отново на фиг. 21.

5.6 Изглед

Изгледът или така-наречения Front-end е изграден с инструментите на ***Angular framework***, показан по-подробно в точка 3.2.4.2, където детайлно е описана и архитектурата на работната рамка, написана на TypeScript. Версията,

[26] Официална документация на Express - https://expressjs.com/

която се използва за проекта е 11.2.6. Първоначално бяха направени опити и с ***ReactJS***, но Angular се оказа по-лек и удобен за ползване. Основата на изгледа е написана във файловете *taskpane.html, taskpane.css и taskpane.ts, намиращи се в src*. В *taskpane.html* е указано основните JavaScript и TypeScript допълнения, които да се включат в проекта, стиловете са описани в *taskpane.css*, а в *taskpane.ts* е извикан и основният инициализиращ метод на разработката на Microsoft –

```
Office.initialize = () => {
  document.getElementById("sideload-msg").style.display = "none";

  // Bootstrap the app
  platformBrowserDynamic()
    .bootstrapModule(AppModule)
    .catch(error => console.error(error));
};
```

Фиг. 22 Инициализация на API

Main.ts ими пояснения, кои модули от ***Angular*** да се използват - @angular/core, @angular/platform-browser-dynamic, ’./app/app.module' и, './environments/environment'.

В src/app се помещават основните парчета код, генериращи изгледа:

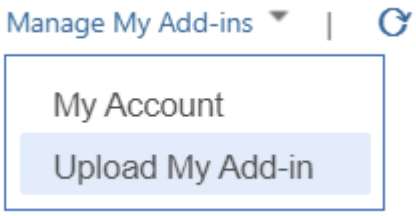
- app-routing.module.ts – настройва маршутизирането между отделните компоненти в уеб приложението.
- app.component.css - стиловете на front-end-а.
- app.component.html – структурата на HTML-а на add-in-а.
- app.component.spec.ts – тестове, които проверяват за успешното създаване на приложението.
- app.module.ts – описание модулите, които се използват в компонентите, спомага за реализирането на dependency injection.
- app.component.ts – асинхронното контролиране на процесите.
- demo.service.ts – в този сървис се конструират заявките към сървъра на *Node.js*.

App.compnent.ts е файлът, който съдържа по-голямата част от логиката. В него всичко е описано асинхронно чрез методите предоставени от Microsoft. Информацията от документа се прихваща в локална променлива, наречена context, който впоследствие се обработва допълнително, за да извлече текста и да подаде заявка с него към съответното място.

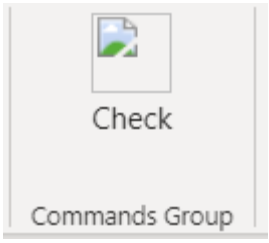
5.7 Добавяне на разширението

Разширението за текстови редактори е разработено както за Microsoft Word, така и за Office 365 Word. За цел тестване, ще се използва Online версията, на която добавянето става по-лесно. След локалното стартиране на проекта през конзолата – cmd -> npm start в директорията на проекта се изчаква да заредят модулите и компонентите. Отваря се Word Online, натиска се Insert -> Add-ins и от Browse опцията се навигира до Manifest.xml файла, на който са описани инструкциите за стартиране и обработване. Впоследствие в дясната част на документа ще се появи прозорец със вече заредения и готов за тестване add-in или разширение.

Фиг. 23 Стъпка 1 за добавяне



Фиг. 234 Стъпка 2 за добавяне



Фиг. 25 Стъпка 3 за добавяне

6 Тестване на разработката

6.1 Задаване на настройките за създаване на средата

За да се настрои средата на тестване са нужни няколко стъпки, които да се изпълнят.

1. Изтегляне на проекта от repository-то. https://github.com/angelbgpl/word-add-in-foreign-words.
2. Трябва да се изтегли Oracle Database Express Edition, която е със стандартна инсталация и сама настройва базата от данни да работи.
3. Наливане на нужните данни в таблиците.
4. Инсталиране на сървърната библиотека на *JavaScript – Node.js*. https://nodejs.org/en/
5. Настройване на *Anglular* да работи локално. https://angular.io/guide/setup-local
6. Изпълняване на командата npm install в командната конзола в папката на проекта, за да може да се инсталират всички node модули.

6.2 Тестване на производителността и извършване на проверки от разработчика

След направата на нужните настроки извършването на проверки ще се изпълни чрез няколко контролни входа:

1. *“Заради нечистото ни съзнание сме на това дередже.”*
2. *“Петър е голям българин и добър човек.”*
3. *“Джойнвам се на всички срещи, на които трябва.”*
4. *“Равенството направи така, че групата бе спечелена от Франция.”*
5. *“Любимото ми занимание на работа е да едитвам стар код.”*

6.2.1Тестване на производителността

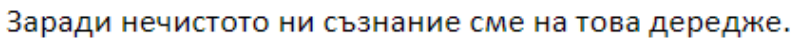
Проведените тестовате позволиха да се направят изводи за системата относно производителността и наличието на грешки в разработката. При стартирането на приложението в конзолата излизат грешки, които понякога зависят и от предоставените от Microsoft средства за писане на add-in разработки.

Фиг. 24 Грешка при стартиране на разширението

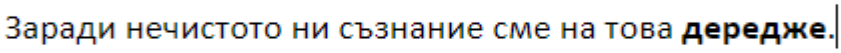
Други забележки, които могат да се имат предвид са дребните детайли по така наречения fron-end или стилът на изгледа, визуализиращ разширението. Като вариант е добавянето на различни видове стилизирания на бутоните и контролите, показващи синонимите.

6.2.2Анализ на конролните проверки

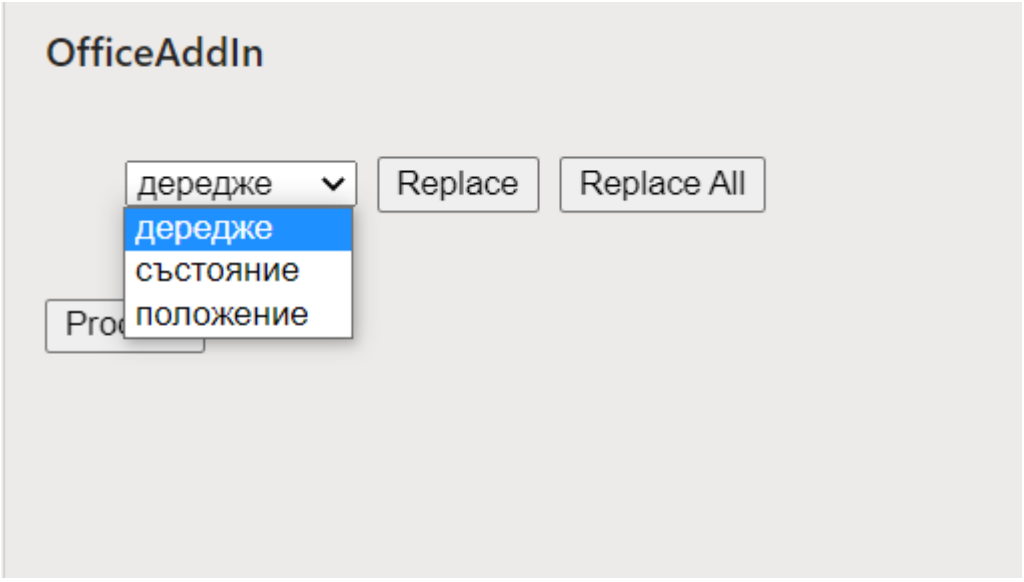
Алгоритъмът, представен в разработката работи успешно, което се вижда на фигурите показано по-долу. Три от примерите са успешни, тоест съдържащи чуждици, а другите два са съставени без чужди думи.



Фиг. 25 Вход за системата - успешен експеримен

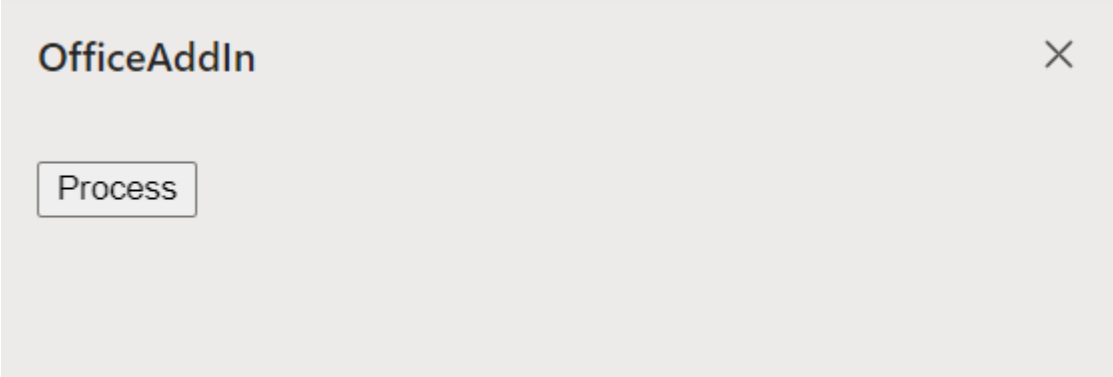


Фиг. 26 Изход от системата с удебелена чуждица



Фиг. 27 Показване на синонимите на чуждицата

На фиг.29 е показан успешният пример, в който думата “дередже” бива удебелена като знак, че тя е чуждица, а в полетата в дясно са показани думите, които могат да я заменят. Чрез избиране на синонима и натискане на бутона *Replace* се заменя чуждата дума с нужната и текстът е опреснен, заедно с полето с контроли, от което биват премахнати бутоните и то бива оставено празно.



Фиг. 28 Показване на контролите, когато в текста не се съдържа чуждица

Пример за изречение, които е *“Петър е голям българин и добър човек.”*. При подаване на този текст като вход на системата отговорът от системата е подобен на този от фиг.30, тоест не показва чуждици за заместване, понеже няма такива.

За останалите три входни изречения резултатите са следните:

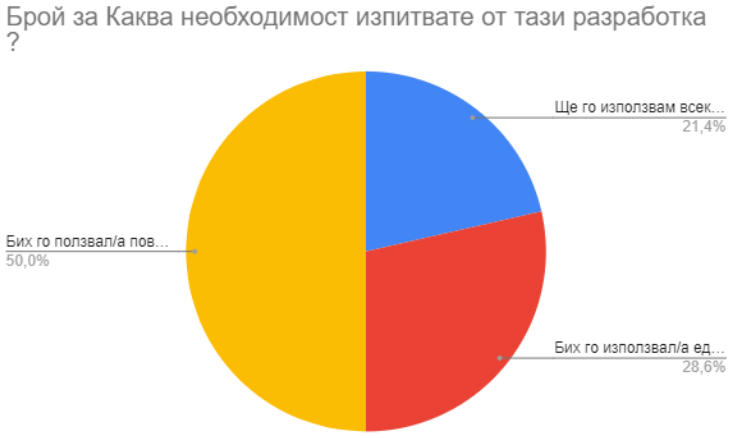
- *“Джойнвам се на всички срещи, на които трябва.”* **Джойнвам се - включвам се, причислявам се, доближавам, съединявам, принаждам, присъствам, взимам участие в.**

- *“Равенството направи така, че групата бе спечелена от Франция.”* **Няма.**
- *“Любимото ми занимание на работа е да едитвам стар код.”* **Едитвам-редактирам.**

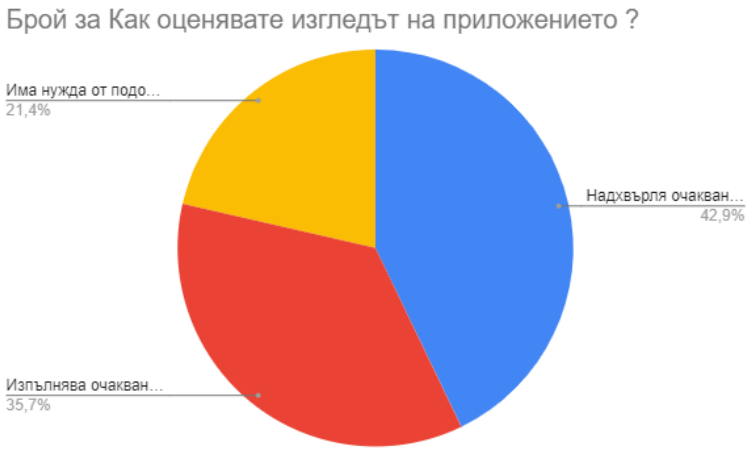
6.3 Тестване от потребители

За целите на тестването от потребители ще се използват свободни входни данни, каквито всеки от тестерите предвиди. Вследствие на използването на разширешието всеки ще попълни анкета в Google Forms, състояща се от няколко въпроса:

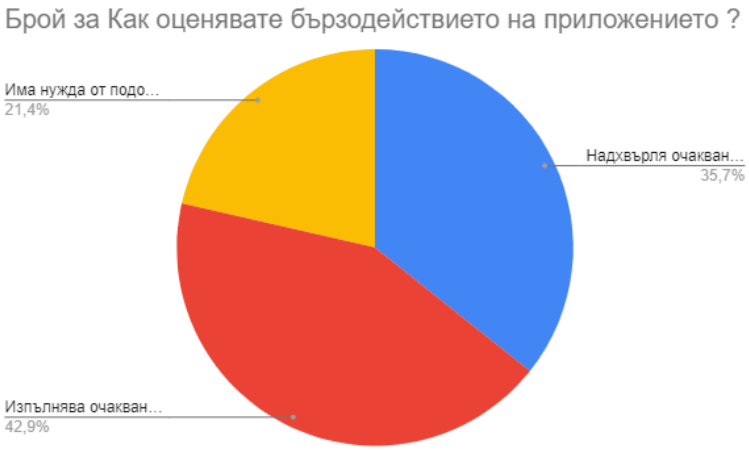
- Вие сте ?
- Вашата възраст е ?
- Каква необходимост изпитвате от тази разработка ?
- Как оценявате изгледът на приложението ?
- Как оценявате бързодействието на приложението ?
- Как оценявате основната функционалност - предлагането на синоним ?
- Какво е впечатлението ви от цялостният начин на работа на добавката ?
- Описание на препоръки или забележки



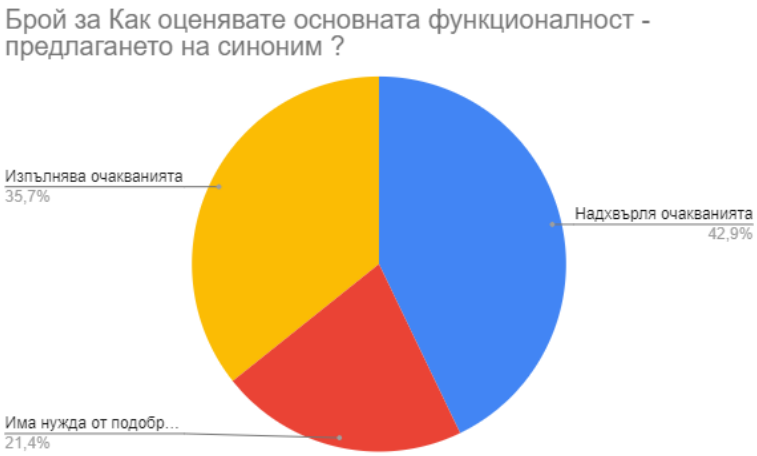
Фиг. 29 Каква необходимост изпитвате от тази разработка ?



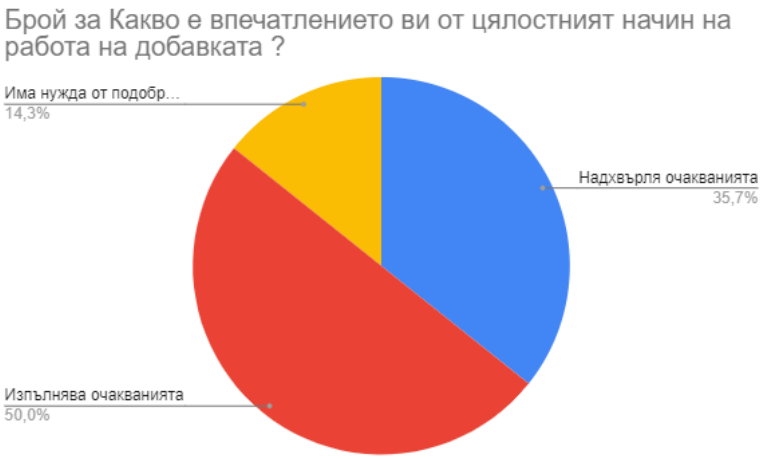
Фиг. 30 Как оценявате изгледът на приложението ?



Фиг. 31 Как оценявате бързодействието на приложението ?



Фиг. 32 Как оценявате основната функционалност - предлагането на синоним ?



Фиг. 33 Какво е впечатлението ви от цялостният начин на работа на добавката ?

На фиг. 29, фиг. 30, фиг. 31, фиг. 32 и фиг. 33 са представени резултатите на въпросите от 3 до 7. В приложение 1 са показани всички резултати в табличен вид. Статистиките показват, че в анкетата са участвали 14 човека, като 10 от тях са работещи, а 4 са се определили като студенти, 8 от тях са между 24 и 34 години, а 6 между 19 и 23 годишна възраст.

Като извод от статистиките и фигурите може да се направи, че първото впечатление от приложението е положително. Трима потребители са оставили и забележки, като един от тях е конкретизирал с препоръка за подобряване на дизайна на контролите.

7 Заключение

В заключение разработката на подобен софтуерен проект е трудна, макар и на пръв поглед доста проста за решение задача. С текущата разработка успях да проектирам *Разширение за текстови редактори за откриване и замяна на чуждици*, като предварително бях отсял различни варианти за изграждане на алгоритъм, изглед, сървърна част, логически слой и бази от данни.

Основното предимство на системата е, че тя създава добър интерфейс, към който лесно може да се закачат други алгоритми, с които тя да се развива. Приложението беше тествано с най-простия алгоритъм с цел да се наблегне на другите му функционалности и да бъдат изгладени детайлите. Други по-сложни алгоритми бяха разгледани като потенциални разширения за бъдещи разработки.

За момента като основни ограничения се посочват заместването на думата с точно определена една, липсата на вариант за заместване на цели изрази и липсата на достатъчно обемен речник в електронен вариант, който да спомогне за по-лесното намиране на чуждите думи в един текст.

С направата на нужните проучвания и тестове като логично продължение на тази система, може да се открият и друго пропуски във функционалността, които да бъдат поправени с нови технологии или по-добри работни рамки.

Езикът е динамична единица, която не спира да се развива. Българският език е познат на света като един от най-трудните езици за научаване, задругяващ със сложността си дори видни полиглоти. Напливът от думи и фрази, които се появяват в езика, вследствие на развитието на технологиите и промяната на мисленето и настройката на хората, понякога може да служи и за обогатяване, но в други случаи “замърсява” езика и го пълни с ненужни думи и фрази, които имат свой еквиваленти на български. Развитието на технологиите в изкуствения интелект, обработката на естестве език и обработката и извличане на данни от текст могат да спомогнат в едно бъдещо развитие на това разширение. С навлизането на все повече новости, решаването на задачата, поставена в началото на този проект изглежда все по-постижимо.

8 Използнава литература и приложения

[1] Веселинов, Димитър. Емпрунтологията в контекста на XXI век. https://www.uni-vt.bg/res/9617/%D0%92%D0%A2%D0%A3_-_%D0%A1%D1%82%D0%BE%D1%8F%D0%BD_%D0%91%D1%83%D1%80%D0%BE%D0%B2_-_%D0%A1%D0%B1%D0%BE%D1%80%D0%BD%D0%B8%D0%BA.pdf

[2] Марина Делева. 10-те най-често използвани чуждици и техните български прабаби. https://www.10te.bg/lyubopitno/10-te-naj-chesto-izpolzvani-chuzhditsi-i-tehnite-balgarski-prababi/

[3] Родопски диалект – речник на туриста в Родопите! https://www.housemilka.com/rodopski-dialekt-rechnik/

[4] Защо чуждиците навлизат в българския език? https://www.tvevropa.com/2010/05/arc27541/

[5] Все повече чужди думи навлизат в речта ни https://novini.bg/bylgariya/obrazovanie/483716

[6] Кимчи, ваканцувам – 9000 нови думи навлезли в българския език. https://www.dnes.bg/obshtestvo/2021/05/25/kimchi-vakancuvam-9000-novi-dumi-navlezli-v-bylgarskiia-ezik.491338

[7] Вероника Лазарова – 20 културни турцизми в българския език. https://www.lifebites.bg/20-kultovi-turcizmi/

[8] *Заети думи в българския език (10)*. (2009, January 20). Българска филология 2008-09. https://bgphilology2008.blogspot.com/2009/01/20090105-10_10.html

[9] Павел Тодоров - Системата за подпомагане на замяната на чуждици с подходящи синоними в текст на български

[10] МАГ-77 - Речник на чуждите думи в българския език. https://knijarnica.bg/product/rechnik_na_chujdite_dumi_v_bylgarskiq_ezik/

[11] An Introduction to N-grams: What Are They and Why Do We Need Them? - https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/

[12] NLP techniques - https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/techniques_word.html

[13] Българския национален корпус - http://metashare.ibl.bas.bg/repository/browse/n-grams-from-bulgarian-national-corpus/e8f38bba6b3311e281b65cf3fcb88b702bab3b7e68234f92aea47df8cdd8a4bf/

[14] What Are Word Embeddings for Text? - https://machinelearningmastery.com/what-are-word-embeddings/

[15] The distributional hypothesis - https://www.researchgate.net/publication/228385506_The_distributional_hypothesis

[16] Jason Brownlee PhD - How to Develop a Word-Level Neural Language Model and Use it to Generate Text. https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/

[17] Krishan. Continuous bag of words (CBOW) – From data to decisions. iksinc.online. https://iksinc.online/tag/continuous-bag-of-words-cbow/

[18] Facebook Inc. (n.d.). Word vectors for 157 languages. fastText. https://fasttext.cc/docs/en/crawl-vectors.html

[19] Wiki word vectors. fastText. https://fasttext.cc/docs/en/pretrained-vectors.html

[20] Bidirectional Encoder Representations from Transformers (BERT) https://en.wikipedia.org/wiki/BERT_(language_model)

[21] Rani Horev - BERT Explained: State of the art language model for NLP. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[22] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[23] Kavita Ganesan - What are N-Grams? https://kavita-ganesan.com/what-are-n-grams/#.YLFPcagzaUk

[24] LibreOffice Extension Development. https://wiki.documentfoundation.org/Development/Extension_Development

[25] What Does Microsoft Office Mean? https://www.techopedia.com/definition/20737/microsoft-office

[26] TheWindowsClub - Microsoft Office vs OpenOffice vs LibreOffice: Which one is better? https://www.thewindowsclub.com/microsoft-office-vs-open-office-vs-libreoffice-which-one-is-better

[27] Sai Teja - What are stop words? https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47

[28] Bulgarian stop words. (n.d.). CountWordsFree. https://countwordsfree.com/stopwords/bulgarian

[29] What Does One-Tier Architecture Mean? - https://www.techopedia.com/definition/17374/one-tier-architecture

[30] Two-Tier Architecture - https://www.techopedia.com/definition/467/two-tier-architecture

[31] N-Tier Architecture - https://www.techopedia.com/definition/17185/n-tier-architecture

[32] Tomas Trajan - Total Guide To Angular 6–11 Dependency Injection — providedIn vs providers:[ ]. https://tomastrajan.medium.com/total-guide-to-angular-6-dependency-injection-providedin-vs-providers-85b7a347b59f

[33] Mahbub Zaman - How To Extract Text From Images Using Tesseract OCR Engine and Python.

https://towardsdatascience.com/how-to-extract-text-from-images-using-tesseract-ocr-engine-and-python-22934125fdd5

Приложения

Приложение 1: Резултати от анкетата, проведена за обратна връзка от използване на приложението.