

**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

[Съдържание 1](#_Toc74576299)

[1 Увод 3](#_Toc74576300)

[1.1 Мотивация за работа 3](#_Toc74576301)

[1.2 Как използването на чуждици може да има негативен ефект върху езика? 3](#_Toc74576302)

[1.3 Основни фактори определящи употребата на чужди думи 4](#_Toc74576303)

[1.3.1 Доколко ни прави впечатление наличието на чужди думи? 4](#_Toc74576304)

[1.3.2 Навлизане на чуждите думи в българския език 4](#_Toc74576305)

[1.3.3 Източници на чуждите думи 5](#_Toc74576306)

[1.3.4 Защо чуждите думи се употребяват толкова често ? 5](#_Toc74576307)

[1.4 Цел на дипломната работа 6](#_Toc74576308)

[1.5 Задачи, произтичащи от целта 6](#_Toc74576309)

[1.6 Структура на дипломната работа 6](#_Toc74576310)

[2 Преглед на областта 6](#_Toc74576311)

[2.1 Използвани термини 6](#_Toc74576312)

[2.2 Съществуващи разработки 6](#_Toc74576313)

[2.2.1 Система за подпомагане на замяната на чуждици с подходящи синоними в текст на български 6](#_Toc74576314)

[2.2.2 Grammarly 6](#_Toc74576315)

[2.2.3 LanguageTool 6](#_Toc74576316)

[2.3 Възможни подходи за разработка 6](#_Toc74576317)

[2.3.1 Възможни алгоритми 6](#_Toc74576318)

[2.3.2 Възможни технологии, нужни за изграждането на разширението 6](#_Toc74576319)

[3 Използвани технологии и езикови ресурси 7](#_Toc74576320)

[3.1 Езикови ресурси 7](#_Toc74576321)

[3.1.1 Речник на чуждите думи в българския език 7](#_Toc74576322)

[3.1.2 Стопдуми в българския език 7](#_Toc74576323)

[3.1.3 Синонимен речник 7](#_Toc74576324)

[3.2 Софтуерни технологии 7](#_Toc74576325)

[3.2.1 База от данни 7](#_Toc74576326)

[3.2.2 Back-end технологии 7](#_Toc74576327)

[4 Проектиране на приложението 8](#_Toc74576328)

[4.1 Защо съм избрал вгражданията на думи? 8](#_Toc74576329)

[4.2 Намиране на подходяща заместваща дума чрез измерване на разстоянието между контекста и думата 8](#_Toc74576330)

[4.2.1 Идея 8](#_Toc74576331)

[4.2.2 Разстояние между два вектора 8](#_Toc74576332)

[4.2.3 Намиране на вграждане на изречение 8](#_Toc74576333)

[4.2.4 Описание на алгоритъма 8](#_Toc74576334)

[4.3 Използване на невронната мрежа, създадена при обучението на вграждания с алгоритъма CBOW 8](#_Toc74576335)

[4.3.1 Идея 8](#_Toc74576336)

[4.3.2 Алгоритъм 8](#_Toc74576337)

[4.4 Използвани технологии и библиотеки 8](#_Toc74576338)

[5 Заключение 8](#_Toc74576339)

[6 Използнава литература 8](#_Toc74576340)

# Увод

## 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 [2] 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, съпорт, лайфстайл, плеймейтка*. Те са описани в статията “*10-те най-често използвани чуждици и техните български прабаби”* на Марина Делева [3].

## 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 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 [4] . 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.

## 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 [5].

### Доколко ни прави впечатление наличието на чужди думи?

**Все повече**[чужди думи](https://novini.bg/biznes/biznes_tehnologii/351199)**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 [6].

### Навлизане 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“ [7]. 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 [8].

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.

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 [7] 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.

### Източници на чуждите думи

От въведението разбрахме, че голяма част от чуждиците и заемките навлизат от английския език, поради наличието на различни фактори, влиящи 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 [9]. Има и други езици, от които сме взаимствали слова [10]:

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

### Защо 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.

## 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.

## 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.

## 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.

# Преглед на областта

## Използвани термини

За пояснение и улеснение тук ще бъдат описани съкращения, термини и означения, които ще се използват по-надолу в текста.

Разширение за текстови редактори за откриване и контекстно-зависима замяна на чуждици ще бъде съкратено на (РТРТЗЧ) . Под целева дума ще се разбира думата, която трябва да бъде заместена със синоним.

## Съществуващи разработки

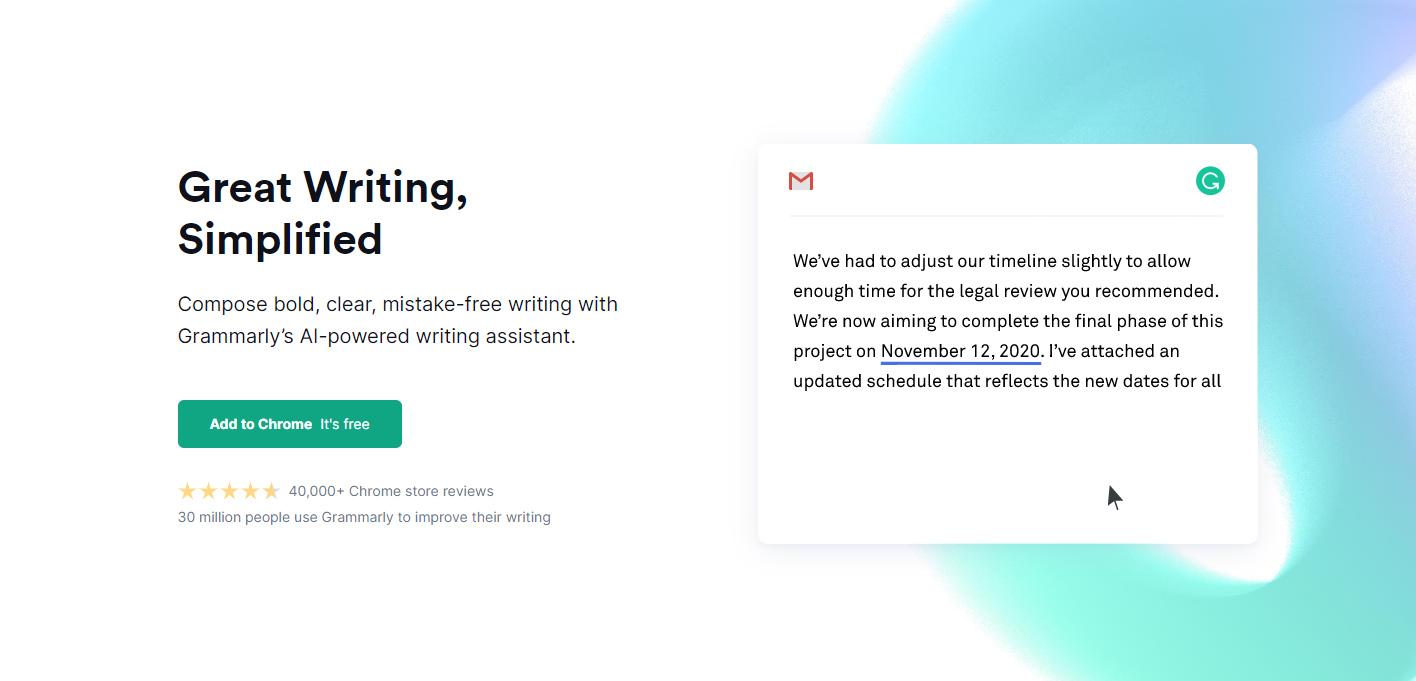
След направеното нужно проучване успех да намеря една система, която да извършва до голяма степен работата, за която е предназначено разширението за текстови редактори и няколко системи със сходни характеристики.

### 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*) [11] 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.

### Grammarly



Grammarly [26] 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 характеристиките, които ние целим да постигнем. 

### LanguageTool

LanguangeTool [27] 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 [28], 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.



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.

## 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 въможните 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.

### Възможни алгоритми

#### Предлагане на заместващ синоним от синонимен речник

##### **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. [11] Речникът на чуждите думи в българския език [12] 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.

##### **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[13], 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*

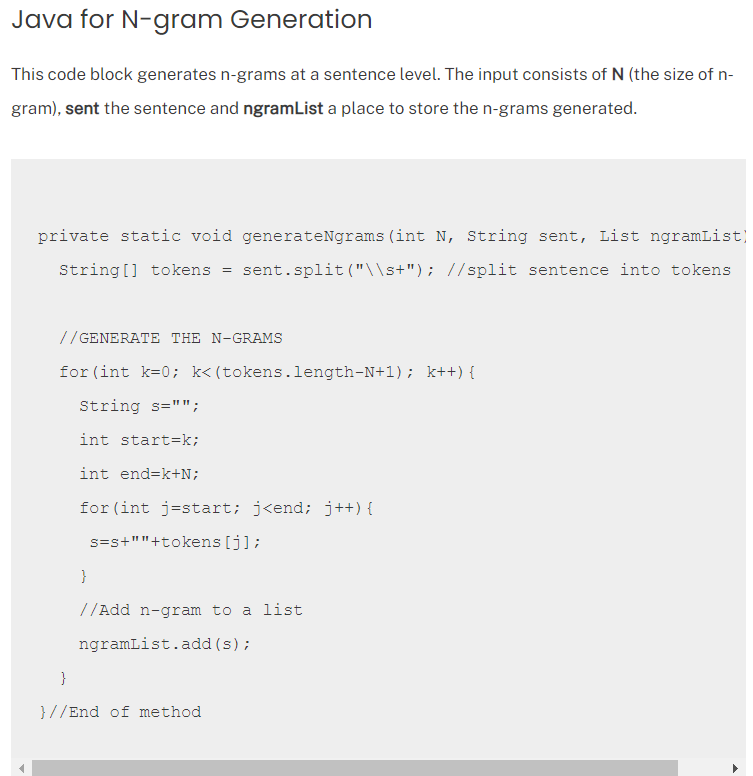
Този прост експеримент доказва, че един от големите недостатъци на алгоритъма е това, че контекстът не се взима предвид. Това е така, защото в този случай избирането на подходяща заместваща дума би се осъществило на случаен принцип. Липсата на отчитане на контекст показва, че проблемът не може да бъде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

#### N-грами

##### **Описание на алгоритъма**

N-грам [14] 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 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 [15] може да се използва силана на N-грамие за решаване на нужната задача.

Съществува и така нареченият Български национален корпус [16], 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.



##### **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.

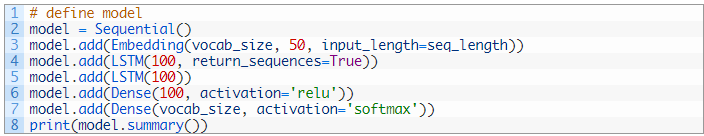
Недостатък на този тип подход е, че за реализацията му е нужно да има предварително изчислени N-гради, както и голямо наличие на информация, с която да се борави по време на работа. В този корпус трябва да се съдържат и рядки думи, тъй като хората, използващи разширението ще има нужда да сменят точно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

#### Изграждане на езиков модел с помощта на вграждания на думи

##### **Описание на алгоритъма**

Вграждане на дума (*word embedding)* [17] 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 [18]. 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 [19], както и Continuous Bag of words и (CBOW) и Continuous Skip-gram Model [20]. 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.

****

За вграждания на български език може да се използват моделите на fasttext [21] или Wiki Word Vectors [22].

##### **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.

#### BERT

##### **Описание на алгоритъма**

Bidirectional Encoder Representations from Transformers (BERT) [23] 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 [24] 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.



##### **Предимства и недостатъци**

Предимствата на BERT[25] 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 избиравето 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.

### 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.

#### OpenOffice

Apache OpenOffice (AOO) [29] 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, серфифицирани 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. [30] 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.

#### LibreOffice

LibreOffice е мощен инструмен [31] 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.

#### 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 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. [33]

За разработката на разширения [34] 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 въху така наречените Word add-ins.

#### 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. [35] 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 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.

# Използвани технологии и езикови ресурси

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

## Езикови ресурси

### Речник на чуждите думи в българския език

Речникът на чуждите думи в българския език[12] 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 предотвати 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.

### Стопдуми в българския език

Стопдумите [36] 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 [37] съдържа 259 думи, които могат да бъдат полезни при обработката на текстовете от разширението за текстови редактори.

### Синонимен речник

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

Синоними[[1]](#footnote-1) 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.

## Софтуерни технологии

В тази подточка ще бъдат представени софтуерните процеси и технологиите за реализирането им.

### База от данни

Базата данни[[2]](#footnote-2) 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.

### Back-end технологии

#### Python

Python[[3]](#footnote-3) 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 e 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 e 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.

##### **Scikit-learn**

Scikit-learn[[4]](#footnote-4) (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.

##### **NumPy**

NumPy[[5]](#footnote-5) 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.

##### **Scrapy**

Scrapy [[6]](#footnote-6)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(),

}

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. 1 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-a 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.

##### **TensorFlow**

TensorFlow [[7]](#footnote-7) е библиотека с отворен код за бързи цифрови изчисления. Тя е създадена и се поддържа от Google и е издадена под лиценза 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.

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.

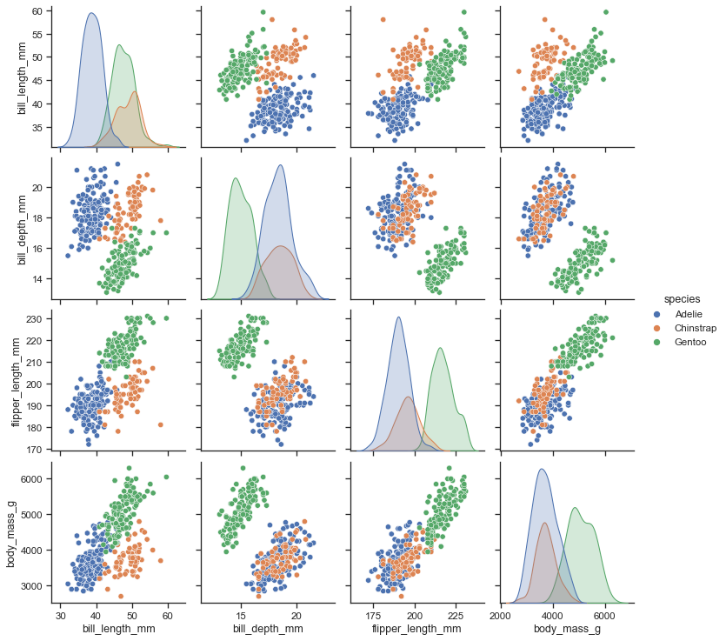
##### **Pandas**

Софтуерната библиотека Pandas [[8]](#footnote-8) 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](https://en.wikipedia.org/wiki/File_format)
* Интегрирана обработка на липсващи данни.
* Преобразуване на [набори](https://en.wikipedia.org/wiki/Data_set) от [данни](https://en.wikipedia.org/wiki/Data_set)
* 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

##### **Seaborn**

Един от най-известните начини за визуалицазия на данни е библиотеката на Python – Seaborn[[9]](#footnote-9). 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. 2 са показани различните видове графики, разработени чрез инструментите на Seaborn.



Фиг. 2 различни видове графики на Seaborn

##### **NLTK**

Една от водещите платформи по създаване на приложения на Python, които да работят с човешки език се казва *NLTK - Natural Language Toolkit*[[10]](#footnote-10). 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 й приложение.

##### **Cx\_Oracle**

Cx\_Oracle[[11]](#footnote-11) 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.

# Проектиране на приложението

## Защо съм избрал вгражданията на думи?

От разгледаните в точка 2.3 подходи за решаване на задачата за замяна на чуждица в текст, съм избрал модела навграждания на думи поради няколко основни предимства, които този модел има над другите описани:

* + 1. Използват се всички думи от контекста за намиране на правилната заместваща дума (а не само тези от лявата страна на чуждицата, както при N-грамите).
    2. Моделите на *fasttext* (описани в точка 2.3.4.2) са тренирани върху много и големи по обем текстове, което ги прави добре обучени. Освен това тези модели съдържат много на брой български думи, включително и редки такива.
    3. Предложените в точка 2.3.4.1 алгоритми, основани на вграждания на думи се справят добре с липсващи в модела думи и са сравнително бързи.
    4. 300-мерните вектори би трябвало да улавят достатъчно добре смисъла на думите.

Важно е да се отбележи, че при търсенето на най-подходяща заместваща дума, ще разглеждам само думи, които са синоними на целевата. Причината е, че най-близките по смисъл думи до дадената целева дума са точно синонимите, така че предложенията, които системата ще връща ще са само синонимите на целевата дума. Това ще спести много изчисления и ще направи алгоритмите по-бързи, като ще остави само най-добрите кандидати за заместващи думи. Това налага и едно ограничение на системата – тя ще може да замества една дума с точно една друга, което означава, че няма да работи, ако искаме да заместваме цели фрази, като например: „от трън та на глог“, „от дъжд на вятър“ и други. Превод на фразеологични изрази не е сред основните цели на системата, но в бъдеще може да се направи странична функционалност, която го прави.

## Намиране на подходяща заместваща дума чрез измерване на разстоянието между контекста и думата

### Идея

Вгражданията надуми са вектори в линейно пространство, като по дефиниция те са научени, така че близки по смисъл думи да имат близки вграждания в линейното пространство. Ако следваме тази логика, можем да представим изречението, в което се съдържа чуждицата като един вектор *s* и множеството от векторите съответстващи на синонимите на чуждицата като *Syn*. След това за всеки синоним изчисляваме разстоянието между него и вектора на изречението и подреждаме получените разстояния в списък (наречен ):

Във формула 10 има две неизвестни неща – как да изчисляваме разстоянието между два вектора и как да намерим вектора s, отговарящ на изречението.

### Разстояние между два вектора

За изчисляване на разстоянието между два вектора можем да използваме познатите от линейната алгебра евклидово и косинусово разстояние, които се дефинират по следния начин:

Нека a и b са два n-мерни вектора. Дефинирам:

*където*

Като мярка за разстояние можем да използваме и скаларното произведение.

### Намиране на вграждане на изречение

Нека е множеството от всички вграждания на думите от входното за системата изречение. Намирането на вектор s, съответстващ на входното изречение, може да се намери по два основни начина:

1. s e медицентърът на векторите, съответстващи на думите от S
2. s e сума на векторите, съответстващи на думите от S, като всеки от тези вектори е утеглен спрямо някакво тегло. Нека са тегла

Формула използваме, ако се водим от допускането, че всяка дума е еднакво важна за контекста на цялото изречение. Това допускане вероятно е вярно за къси изречения, но няма гаранция, че ще даде добри резултати при дълго изречение. Тъй като повечето от изреченията в анотираните данни са къси, формула 16 би трябвало да даде добри резултати.

Премахването на стопдуми би трябвало да премахне от S тези думи, които ще отклонят вектора s от контекста на самото изречение. Важно е да се отбележи, че S съдържа и чуждицата, която ще трябва да бъде заместена, защото нейното значение е от изключително голяма (дори най-голяма) важност за смисъла на S.

Формула 17 предлага различните думи да се утеглят различно, като най-логично е думите, които са най-близки до целевата дума да са с най-голяма тежест, а далечните – с най-малка. В частност – самата целева дума трябва да има най-голяма тежест. Предлагам следните методи за изчисляване на описаните тежести:

Тежестите ще илюстрирам с чуждицата *„****абсолютен”*** и примерното изречение:

Кралят е ***абсолютният*** господар на империята си.

* Линейни тегла - Този метод дава тежест 1 на най-отдалечената от чуждицата дума, 2 на тази, която е с една дума по-близка и така нататък, докато самата чуждица получи най-голяма тежест:

**фиг. 9** *Пример за линейно утегляване на думите в изречението „Кралят е абсолютният господар на империята си” с чуждица „абсолютен”*

Теглата лесно се изчисляват по следния алгоритъм:

1. Нека *sentenceSize* е броят думи в изречението, *maskedIndex* е индексът на чуждицата, а *weights* е списъкът с теглата
2. *maxWeight = max(sentenceSize - maskedIndex, maskedIndex + 1)*

*maxWeight* е теглото на чуждицата (максималното тегло измежду всички), което е с 1 по-голямо от разстоянието от чуждицата до най-отдалечената от нея дума

1. Изчисляваме левите тегла – тези от началото на изречението до чуждицата

**int** currentWeight = maxWeight - maskedIndex;

**for** (; currentWeight < maxWeight; currentWeight++)

weights.add(Double.*valueOf*(currentWeight));

1. Изчисляваме десните тегла – тези от чуждицата до края на изречението

currentWeight = maxWeight;

**int** weigthsToBeAdded = sentenceSize - weights.size();

**for** (; weigthsToBeAdded > 0; weigthsToBeAdded--)

weights.add(Double.*valueOf*(currentWeight));

* Квадратични тегла – имат сходна идея с линейните. Целта им е близките до целевата дума да имат още по-голяма тежест, а далечните – още по-малка спрямо линейните тегла. Изчисляват се като се взимат квадратите на линейните тегла. Ако линейните тегла бележим с формулата „х”, то квадратичните можем да отбележим с „х2”. Аналогично можем до имаме „2х + 1” „х3” и други подобни.

**фиг. 10** *Пример за квадратично утегляване на думи в изречението „Кралят е абсолютният господар на империята си” с чуждица „абсолютен”*

Теоретично могат да се използват и много други по-бързорастящи функции. Тъй като изреченията в анотираните данни са кратки, при използването на такива функции ще се стигне до положение, в което целевата дума да има почти цялата тежест, което обезсмисля контекста. Поради тази причина ще използвам главно линейни и квадратични тегла.

При по-дълги изречения дори може да се приложи премахване на твърде отдалечени думи, т.е утегляването им с 0. Това не се налага при анотираните данни, които както казахме, съдържат основно къси изречения.

### Описание на алгоритъма

Предлагам следния алгоритъм:

1. Вход: (целева) дума и контекст (изречение, което е разделено на множество от думи)
2. Намираме вгражданетона входната дума и на всяка дума от входното изречение, които обозначаваме съответно с *w* и *S*.
3. За всеки синоним на входната дума намирам съответното му вграждане и го добавям в множеството *Syn*
4. Намирам вектора *s*, който съотвества на изречението *S*, по един от начините описан в точка 4.2.3
5. За всеки syn от *Syn* изчислявам:

*,*

като *dist* e функция за намиране на разстояние между 2 вектора, която една от трите – евклидово разстояние, косинусово разстояние, скаларно произведение (виж точка 4.2.2).

Ако целевата дума я няма в синонимния речник, тогава се изчислява за всяка дума.

1. Сортирам списъка във възходящ ред (най-напред в списъка са най-близките до входното изречение синоними)
2. Връщам списъка *propolsals*

## Използване на невронната мрежа, създадена при обучението на вграждания с алгоритъма CBOW

### Идея

Идеята зад следващия алгоритъм е да използвам невронната мрежа, създадена при обучението на алгоритъма за предсказване на липсваща дума в текст *CBOW (Continuous Bag of Words)*. Ще използвам означенията от точка 2.3.4.1, които бяха следните:

V – брой думи в речника, който моделът използва  
N – брой измерения, които ще имат получените вграждания  
C – брой думи от контекста на целевата дума  
X1 X2 … XC – V-мерни входни вектори – това са векторите отговарящи на думите от контекста  
X – вектор, който е средноаритметичен на входните вектори  
H = (h1 h2 … hN) – N – мерен вектор, който е изход от скрития слой  
Y = (y1 y2 … yV) – V-мерен изходен вектор  
WI - входна матрица (input matrix) с размерност VxN. i-тия ред от нея (за всяко i от 1 до V) представлява наученото *input* вграждане (входно вграждане, input word embedding) на i-тата дума  
WO - изходна матрица (output matrix) с размерност NxV. i-тия стълб от нея (за всяко i от 1 до V) представлява наученото (output \ изходно) вграждане на i-тата дума

Алгоритъмът *CBOW*, който научава (генерира) вграждания на думи, се използва за предсказване на липсваща дума в текст, въз основа на *контекста* ѝ. Векторът *Y’* съдържа тези предсказания, като той се изчислява по следните формули:

(7)

При решаването на задачата за замяна на чуждица в текст ще използвам описания в точка 2.3.4.2 модел, който е научил вграждания за български език с алгоритъма *CBOW*.

Ако събирането във формула 7 се извърши на по-късен етап, тя може да се представи по следния начин:

,

Във формули 18 и 8 могат да се направят няколко опростявания. Първо, векторите са *one-hot encoded*, т.е. съдържат 1 на i-та позиция и нули на всички останали. Това означава, че за произволно i от 1 of C, ще е точно i-тия ред на матрицата WI, което е точно i-тия *input* вграждащ вектор Hi.

Входниявграждащ вектор (*input embedding vector*) за цялото изречение можем да изчислим спрямо някое от утегляванията описани в точка 4.2.3.

Второ, матрицата *WO* съдържа всички научени вграждания, на нас обаче ни трябват само вгражданията на синонимите на целевата дума, което означава, че можем да намалим размерността на *WO* от *NxV* до нова матрица с размерност *N х |Syn|* (*Syn* е множеството от всички синоними на целевата дума), оставяйки само вгражданията на синонимите.

При това опростяване, резултатният вектор *Y* ще има само *C* на брой координати, като след утегляване на *H* със *softmax*, i-тата координата ще отговаря на вероятността i-тият синоним да е заместващият. Тъй като в случая е достатъчно да подредим тези координати по-големина, приравняването им към вероятности не е необходимо и няма да се прави.

От формула 21 и правилото за умножение на матрици се вижда, че i-тата координата на *Y* е точно скаларното произведение на *H* и i-тия стълб на *WO* (който е вграждане на съответния синоним).

### Алгоритъм

При използването на невронната мрежа на *CBOW* с направените промени, полученият алгоритъм е почти същият, както описаният в точка 4.2.4 с две разлики:

* В точка 4, *s* се намира чрез утегляване на *input embeding* векторите за съответната дума от изречението, а не вгражданията (които са *output embedding* векторите )
* В точка 5, разстоянието, което се използва е само скаларно произведение.

## Използвани технологии и библиотеки

Системата за замяна на чуждици е реализирана за езика за програмиране *Java*, а част от обработката на данните е на *Python*. С цел по-бързото изчисляване на резултатите от прилагането на избраните алгоритми върху анотираните данни съм направил следните оптимизации:

1. Извлякъл съм всички думи от анотираните данни, заедно със синонимите на целевите думи в данните във файл *Words\_in\_anotated\_data.txt*
2. За моделите *CBOW Word Vectors* и *Wiki Word Vectors* съм извлякъл файлове, които съдържат вгражданията за съответния модел само на думите от анотираните данни.
3. Входнитевграждащи вектори за *CBOW Word Vectors* и вгражданията за *CBOW Word Vectors* и *Wiki Word Vectors* са извлечени от библиотеката *fasttext* на *Python*, която работи с описаните в точка 2.3.4.2 модели. Вгражданията на *Wiki Aligned Vectors* съм взел директно от файла на сайта на *fasttext.*
4. Всички вграждания, както и синонимния речник съм добавил в база от данни, за да може системата да има бърз достъп до тях.

Освен стандартните *Java* библиотеки, съм използвал и библиотеката *JDBC* за работа с бази от данни. В *Pyhton* съм използвал библиотеката *fasttext*. Графиките са правени с *Excel*.

# Заключение

Проектирах система за замяна на чуждици, която използва алгоритмите, свързани с вместени думи, като изпробвах различни варианти на модификации върху тези алгоритми, както и различни набори от вектори. Основно предимство на използваните модели е, че те са тренирани върху много български текстове, съдържат вмествания за много български думи и се справят добре с липсващи думи. Получените резултати са много добри и системата има потенциал за разширяване. Основно ограничение е, че реализираната система замества една целева дума с точно една друга. В някои случаи може да се наложи заменяне на една дума с две (или повече) или обратното – заменяне на много думи с една. Както споменахме, при заменяне на една дума с точно една друга дума, системата не улавя значението на по-сложни фрази. Този проблем може да се разреши като системата проверява дали посочената от потребителя дума, заедно със съседните ѝ образуват фразеологичен израз, като за целта системата ще трябва да разполага с речник на фразеологичните изрази. Достатъчно обемен речник на чуждите думи пък ще позволи такива думи да се разпознават автоматично в даден текст и да се заместват със синоним от предложенията по желание на потребителя.

Друго разширение е свързано с думите на английски език. Както споменахме, много потребители често се сещат първо за английската дума. За замяна на английска дума, системата може да провери всички нейни преводи и да процедира с тях като с нейни синоними, като приложи над тях същите алгоритми с вмествания на думи, които са описани в точка 4. Друго предложение за бъдещо развитие на системата се отнася до дългите текстове. Ако текстът е достатъчно обемен, може да се намери общата му тематика и това да помогне за по-доброто откриване на значенията на поисканите от потребителя думи.

Описаният в точка 5.3.5 недостатък, свързан с предпочитанието на системата да избира заместващи думи със същия корен като целевата, може да се преодолее, ако съответните заместващи думи бъдат целенасочено поставени по-ниско в предложенията или разстоянието между контекста и съответната заместваща дума изкуствено се увеличи.

С цел по-бързата работа на системата, може да се тества дали вектори с размерност по-малка от 300 (например 150 или 100), ще се правят еднакво добре, колкото и 300-измерните с предлагането на подходящи синоними.

Всеки език е динамичен и продължава да се развива. Българският не е изключение и в него също се наблюдава засилена поява на нови думи, които се появяват в следствие на развиващото се както технологично, така и като психология и начин на мислене общество. Все по-трудно е да се намери балансът (или казано на български равновесието) между необходимостта от навлизане на нови думи и стремежът за опазване на чистотата на нашия език. Предизвикателство остава и разбирането на значенията и смисъла на думите, както от хората, така и от компютрите. С навлизането на все повече езикови модели и нови алгоритми, системи като тази замаяна на чуждици ще се справят все по-добре с разбирането на естествения език и ще ни помогнат и ние да го разбираме и оценим по-добре.

# Използнава литература

[1]Веселинов, Димитър. [Емпрунтологията в контекста на ХХІ век](http://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). <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] [https://rechnik.chitanka.info/w/%D0%B7%D0%B0%D0%B5%D0%BC%D0%BA%D0%B](https://rechnik.chitanka.info/w/%D0%B7%D0%B0%D0%B5%D0%BC%D0%BA%D0%25B)0

*[3] 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/***](https://www.10te.bg/lyubopitno/10-te-naj-chesto-izpolzvani-chuzhditsi-i-tehnite-balgarski-prababi/)

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

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

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

[7] 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>

[8] 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>

[9] Вероника Лазарова – 20 култови турцизми в българския език. <https://www.lifebites.bg/20-kultovi-turcizmi/>

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

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

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

[13] [www.slovored.com.https://slovored.com/search/synonymous/%D1%86%D0%B5%D0%BB](http://www.slovored.com.https://slovored.com/search/synonymous/%D1%86%D0%B5%D0%BB)

[14] 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/>

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

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

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

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

[19] 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/

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

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

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

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

[24] 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>

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

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

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

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

[29] Apache OpenOffice. https://en.wikipedia.org/wiki/Apache\_OpenOffice

[30] OpenOffice Wiki. <https://wiki.openoffice.org/wiki/Extensions_development>

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

[32] LibreOffice Extension Development. <https://wiki.documentfoundation.org/Development/Extension_Development>

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

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

[35] 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>

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

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

1. Синоними - http://rechnik.info/%D1%81%D0%B8%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC [↑](#footnote-ref-1)
2. Какво е база от данни? https://delc2.fmi.uni-plovdiv.net/courseCategory/show/8 [↑](#footnote-ref-2)
3. За какво се използва Python и какви са ползите му? https://blog.superhosting.bg/why-and-for-what-python.html [↑](#footnote-ref-3)
4. Официална документация на Scikit-learn - https://scikit-learn.org/stable/getting\_started.html [↑](#footnote-ref-4)
5. Официална документация на NumPy - https://numpy.org/doc/stable/user/whatisnumpy.html [↑](#footnote-ref-5)
6. Официална документация на Scrapy - https://docs.scrapy.org/en/latest/index.html [↑](#footnote-ref-6)
7. Introduction to the Python Deep Learning Library TensorFlow - https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/ [↑](#footnote-ref-7)
8. Официална документация на Pandas - https://pandas.pydata.org/about/index.html [↑](#footnote-ref-8)
9. Документация за Seaborn - https://seaborn.pydata.org/ [↑](#footnote-ref-9)
10. Официална документация на NLTK - https://www.nltk.org/ [↑](#footnote-ref-10)
11. Сайт на cx\_Oracle - https://cx-oracle.readthedocs.io/en/latest/ [↑](#footnote-ref-11)