

**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 2](#_Toc72961786)

[1 Увод 5](#_Toc72961787)

[1.1 Същност на понятието чужда дума. С какво се различава заемката от чуждицата? 5](#_Toc72961788)

[1.2 Как използването на чуждици може да има негативен ефект върху езика? 5](#_Toc72961789)

[1.3 Основни фактори определящи употребата на чужди думи 6](#_Toc72961790)

[1.3.1 Доколко ни прави впечатление наличието на чужди думи? 6](#_Toc72961791)

[1.3.2 Навлизане на чуждите думи в българския език 6](#_Toc72961792)

[1.3.3 Източници на чуждите думи 7](#_Toc72961793)

[1.3.4 Защо толкова често използваме чужди думи ? 7](#_Toc72961794)

[1.4 Цел на дипломната работа 8](#_Toc72961795)

[1.5 Задачи, произтичащи от целта 8](#_Toc72961796)

[1.6 Проблем с разпознаване на значението на дума 9](#_Toc72961797)

[1.7 Структура на дипломната работа 9](#_Toc72961798)

[2 Съществуващи разработки 10](#_Toc72961799)

[2.1 Основни понятия 10](#_Toc72961800)

[2.2 Съществуващи разработки 10](#_Toc72961801)

[2.2.1 Twinword Writer 10](#_Toc72961802)

[11](#_Toc72961803)

[2.3 Възможни алгоритми 12](#_Toc72961804)

[2.3.1 Задача за предсказване на липсваща дума в текст като генерализация на задачата за заменяне на чуждица 12](#_Toc72961805)

[2.3.2 Заместване на чуждицата с произволен синоним от синонимен речник 13](#_Toc72961806)

[2.3.3 N-грами 14](#_Toc72961807)

[2.3.4 Вграждания на думи (Word embeddings) 16](#_Toc72961808)

[2.3.5 BERT 19](#_Toc72961809)

[2.3.6 Сравнение между алгоритмите 20](#_Toc72961810)

[3 Използвани езикови ресурси 21](#_Toc72961811)

[3.1 Синонимен речник 21](#_Toc72961812)

[3.2 Тълковен Речник 22](#_Toc72961813)

[3.3 Анотирани данни 23](#_Toc72961814)

[3.3.1 Изисквания към анотираните данни 23](#_Toc72961815)

[3.3.2 Извличане на анотираните данни 23](#_Toc72961816)

[3.3.3 Важни статистики за анотираните данни 25](#_Toc72961817)

[3.4 Речник на чуждиците в българския език 28](#_Toc72961818)

[3.5 Списък със стопдуми за български език 28](#_Toc72961819)

[4 Метод за предлагане на замени на чуждици в българския език 29](#_Toc72961820)

[4.1 Защо съм избрал вгражданията на думи? 29](#_Toc72961821)

[4.2 Намиране на подходяща заместваща дума чрез измерване на разстоянието между контекста и думата 29](#_Toc72961822)

[4.2.1 Идея 29](#_Toc72961823)

[4.2.2 Разстояние между два вектора 30](#_Toc72961824)

[4.2.3 Намиране на вграждане на изречение 30](#_Toc72961825)

[4.2.4 Описание на алгоритъма 32](#_Toc72961826)

[4.3 Използване на невронната мрежа, създадена при обучението на вграждания с алгоритъма CBOW 33](#_Toc72961827)

[4.3.1 Идея 33](#_Toc72961828)

[4.3.2 Алгоритъм 34](#_Toc72961829)

[4.4 Използвани технологии и библиотеки 34](#_Toc72961830)

[5 Експерименти за оценка на разработения метод 36](#_Toc72961831)

[5.1 Метрики за оценяване на системата 36](#_Toc72961832)

[5.1.1 Ограничено и разширено множество от контекстови синоними 36](#_Toc72961833)

[5.1.2 Описание на метриките 36](#_Toc72961834)

[5.2 Постановка на експеримента 37](#_Toc72961835)

[5.3 Анализ на резултатите 38](#_Toc72961836)

[5.3.1 Резултати от алгоритъма намиране на най-близък синоним до контекста без утегляване на думите от контекста 38](#_Toc72961837)

[5.3.2 Резултати от алгоритъма намиране на най-близък синоним до контекста с утегляване на думите от контекста 40](#_Toc72961838)

[5.3.3 Резултати от алгоритъма, който използва невронната мрежа на CBOW 42](#_Toc72961839)

[5.3.4 Анализ на най-добрите резултати от приложените алгоритми 42](#_Toc72961840)

[5.3.5 Анализ на причините, които влияят на успеваемостта на системата 46](#_Toc72961841)

[6 Заключение 50](#_Toc72961842)

[7 Библиография 51](#_Toc72961843)

# Увод

## Същност на понятието чужда дума. С какво се различава заемката от чуждицата?

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" \t "_blank)**навлизат в речта ни. Основно те са английски, свързани са с новите технологии и с глобалната 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.

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]:

* международни - *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.

## Цел на дипломната работа

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. 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 изгради мост в общуването между генерациите и да се спомогне “изчистването” на езика от ненужните думи и фрази. 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.

# Съществуващи разработки

## Основни понятия

текст

## Съществуващи разработки

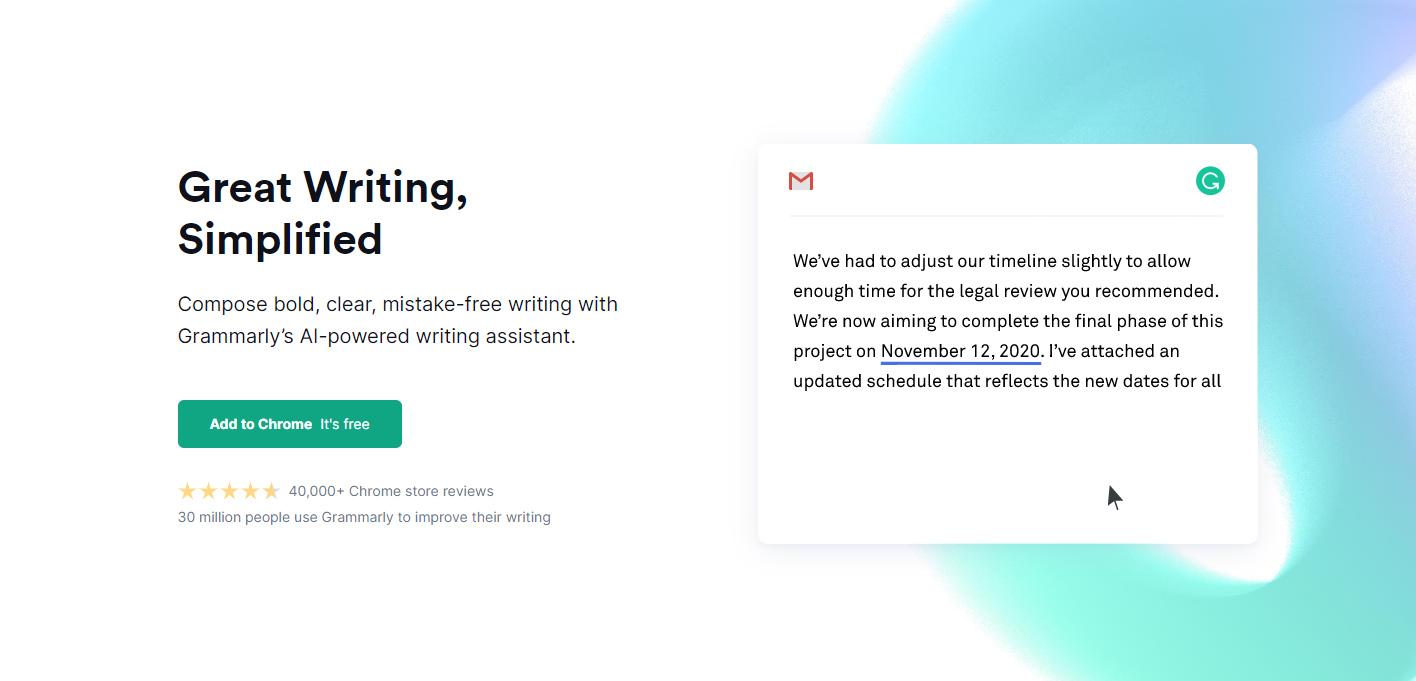
След направеното нужно проучване успех да намеря една система, която да извършва до голяма степен работата, за която е предназначено разширението за текстови редактори и няколко системи със сходни характеристики.

### 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 [12] 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 [13] 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 [14], 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

Тази 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

#### Задача за предсказване на липсваща дума в текст като генерализация на задачата з а заменяне на чуждица

Проблемът със заместването на чуждица с подходящи синоними спрямо контекста е много специфичен и не успях да намеря друга система, която изпълнява точно тази задача. Има обаче много разработки, свързани със задачата за предвиждане / запълване на липсваща дума в изречение, която е генерализация на поставената. Постановката на общата задача е следната – дадено е изречение, в което една от думите липсва (е маскирана). Целта е да се открие най-вероятната липсваща дума. Важно е да се отбележи каква е разликата между задачата за предсказване на липсваща дума и тази за заместване на чуждица. При втората има допълнителна информация, а именно, че липсващата дума е синоним на дума, която е известна за потребителя и системата. Тази разлика ще демонстрирам със следния пример, в който липсващата дума е отбелязана със символа „\_“ :

Задача за предсказване на липсващата дума (Задача 1)  
(по-обща задача)  
Изречение: Кралят е \_ господар на империята си.

Задача за замяна на чуждица (Задача 2)  
(по-специфична задача)  
Изречение: Кралят е \_ господар на империята си.  
Дума, която искаме да заместим (чуждица): абсолютен

Възможните запълващи думи при поставена задача 1 могат да са много и най-разнообразни, например: добър, лош, отговорен, безотговорен, независим, самодържавен и така нататък. В задача 2 наборът от възможни думи е много по-малък, защото съдържа само синонимите или сходните думи на чуждицата *абсолютен*. Такива са например думите върховен, самодържавен, независим, деспотичен, тираничен и така нататък. Задача 1 цели да намери дума, която съответства на празното място както синтактично, така и семантично, но по-скоро набляга на синтактичната част, защото приема думи с разнообразни и дори противоречащи си значения. Задача 2 търси дума, която съответства семантично на контекста, а що се отнася до синтаксиса – за заместващата дума е ясно, че трябва да е от същата част на речта като липсващата, което ще бъде изпълнено автоматично, ако за запълваща дума използваме неин синоним.

В точки 2.3.3 до 2.3.5 ще разгледам именно алгоритми, които се опитват да решат задачата за запълване / предсказване на липсваща дума в изречение, тъй като те са добра отправна точка за решаването задачата за заменяне на чуждица в текст. При разглеждането на алгоритмите ще взема и предвид това кой алгоритъм ще може да се нагоди по-лесно спрямо допълнителната информация, която е известна в задача 2. Преди това обаче ще разгледам един по-прост алгоритъм.

#### Заместване на чуждицата с произволен синоним от синонимен речник

##### **Описание на алгоритъма**

Използването на синонимен речник може значително да улесни намирането на заместваща дума за дадена чуждица. Синонимният речник на *БГ ОФИС* [15] съдържа 22889 думи, за всяка от които са посочени разнообразни синоними (речникът ще бъде описан по-подробно в точка 3 – използвани езикови ресурси). Наивният алгоритъм за заменяне на чуждица е следният: намират се синонимите на чуждицата от синонимния речник и те се предлагат като заместващи думи на чуждицата. При този алгоритъм не е ясно как точно да се подредят предложените заместващи думи. Това може да стане по няколко начина - на случаен принцип, да са подредени спрямо честотата си на използване в български език или да са подредени спрямо реда, в който са в речника.

##### **Предимства и недостатъци на алгоритъма**

Този алгоритъм има няколко предимства. Най-важното от които е бързината му – извличането на всички синоними на дадена дума от речника е бърз процес, ако речникът е във вид на текстов файл и още по-бърз, ако синонимите са в база от данни. Допустим е и вариантът – при стартиране на системата, тя да изчете всички думи от речника, заедно със синонимите ѝ и да ги запамети в структура от данни за бърз достъп, като например *HashMap*, което е възможно и поради малкия обем памет, който заема синонимният речник. Алгоритъмът би работил добре, ако чуждицата има много малък брой синоними, например по-малко от 3-4, като в този случай подредбата им не би била от голямо значение.

В повечето случаи обаче, произволна дума би имала много повече синоними. Нека да разгледаме синонимите на чуждицата абсолютен в синонимния речник на *БГ ОФИС* [15]:

абсолютен  
(прил.) положителен, истински, неоспорим, безспорен  
(прил.) установен, категоричен, определен, сигурен, точен, изричен, ясен, безусловен, обезателен, формален  
(прил.) повелителен, заповеден, заповеднически, рязък  
(прил.) неограничен, пълен, самовластен, тираничен, деспотичен, автократичен  
(прил.) върховен, независим, суверенен  
(прил.) решителен, окончателен  
(прил.) отявлен, явен, флагрантен, същински, същи, чист  
(прил.) безрезервен  
(прил.) цялостен, целокупен  
(прил.) самодържавен  
(прил.) съвършен, идеален, цял, безукорен, завършен  
(прил.) очевиден

Думата абсолютен има цели 45 синонима в различните си значения. Очевидно е, че ако синонимите бъдат подредени на случаен принцип, няма никаква гаранция, че най-подходящите заместващи думи ще бъдат сред първите в списъка. Подреждането на предложенията по честота на използване би помогнало, ако е вярно допускането, че по-често използвана дума да е по-вероятно да бъде подходяща заместваща за чуждицата. Остава обаче проблемът, че този алгоритъм по никакъв начин не отчита контекста, в който чуждицата е използвана, което поставя под съмнение ефективността му. Без да отчита контекста, алгоритъмът няма как да разреши проблема с разпознаването на значението на дума, описан в точка 1.6.

#### N-грами

##### **Описание на алгоритъма**

N-грам [16] наричаме последователност от N думи. Например, последователностите „добър човек“, „аз съм“, „обичам храната“ са 2-грами, наричани още биграми. Примерни 3-грами са „много добър човек“ и „той обича хората“. Аналогично има 4-грами, 5-грами, 6-грами и така нататък. N-грамите могат да се използват при решаването на проблема / задачата за предсказване на следваща дума в текст на база на предишните. Тази задача е по-специфичен случай на задачата за предсказване на липсваща дума в текст, описана в точка 2.3.1, като в тази по-специфична задача липсващата дума задължително е последната.

Един подход за решаването на задачата за предсказване на следващата дума е вероятностният. Той разчита на голям корпус от текстове на съответния език. Нека от сега нататък *w1 w2 … wn* е последователност от *n* думи (n >= 1), а *х* е дума. Да въведем следните означения:

C(w1  w2  … wn) – брой срещания на последователността от думи w1  w2  … wn в корпуса

P (x|w1 , w2 , … , wn) – вероятността думата x да е следващата след последователността от думи w1 w2 … wn. Това може да се счита за съкратен запис на P (X = x | Y1 = w1 , Y2 = w2 , … , Yn = wn), където X и Yi са случайни променливи, като X е стойността / съдържанието на следващата дума, а Yi – стойността / съдържанието на (n - i + 1) - тата предишна дума.

P(w1 , w2 , … , wn) – съкращение на P (X1 = w1 , X2 = w2 , … , Xn = wn), където X1, X2, … Xn са случайни променливи; обозначава вероятността произволна последователност от n символа да е точно w1 , w2 , … , wn

w1i = w1 , w2 , … , wi

Можем да изчислим вероятността ***wn*** да е следваща дума в изречението ***w1  w2  … wn-1*** въз основа на дадения корпус по следния начин:

(1)

Тази сметка може да се направи за всяка дума ***wn*** от корпуса и след това да се вземе думата с най-голяма вероятност.

Проблемът с този подход е, че броят на изреченията ***w1  w2  … wn-1*** и ***w1  w2  … wn-1 wn***в корпуса може да е много малък (дори равен на нула, особено ако ***n*** е голямо число), при което изчисленията стават изключително неточни, а дори може да стигне до деление на 0.

N-грамите предлагат решение на проблема. На всеки N-грам ***w1 , w2 , … , wn*** може да се съпостави вероятността за *n > 1* и за *n = 1*. Нека да приемем, че *N = 2.* Тогава*,* при изчисляването наще използваме само последната дума преди (т.е последната преди липсващата, която е ) по следния начин:

В тази формула използваме допускането, че вероятността да срещнем дадена дума в текст зависи само от думата преди нея (а не от всички думи преди нея), което се нарича ***допускане на Марков*** [16]. Вероятността може да се сметне много лесно, като се използва корпусът от изречения:

Аналогични са и формулите за произволни N-грами (т.е за произволно N), тогава можем да приложим формулите:

Най-вероятната дума е тази , за която е най-голямо. Тъй като е цяло положително число и не зависи от , можем да го изпуснем от сметките и да намерим най-вероятната дума изчислявайки израза:

(6)

##### **N-грами за български език**

Българският национален корпус [35] съдържа 1-грами, 2-грами, 3-грами, 4-грами и 5-грами за български език за около 1,2 милиарда думи. Моделът съдържа както N-грами за думи в основна форма, така и N-грами за думите заедно с различните им словоформи.

##### **Предимства и недостатъци на алгоритъма**

Алгоритъмът за намиране на следваща дума в текст, основан на N-грами има няколко предимства. Първо, ако използваме малко N, например 2, 3, 4 и имаме предварително изчислени N-грами, то изчисляването на съответните вероятности би станало сравнително бързо. Освен това, лесно може да се направи модификация на формулите, за да се нагодят те към задачата за заменяне на чуждица. Модификацията се състои в това, че вместо се разглежда всяко от корпуса (формула 6), можем да разгледаме само тези , които са синоними на чуждицата, като синонимите вземем от речник. Това ще намали допълнително изчисленията и ще ускори алгоритъма още повече.

Недостатък на вероятностния алгоритъм, основан на N-грами е, че той разчита на голям корпус от данни и предварително изчислени N-грами. Корпусът трябва да включва и рядко срещани думи, защото често потребителите ще искат да заменят именно такива. Друг голям недостатък е, че този подход по никакъв начин не отчита думите, които са след чуждицата, а именно те може да се окажат ключови за това коя да е най-подходящата замяна.

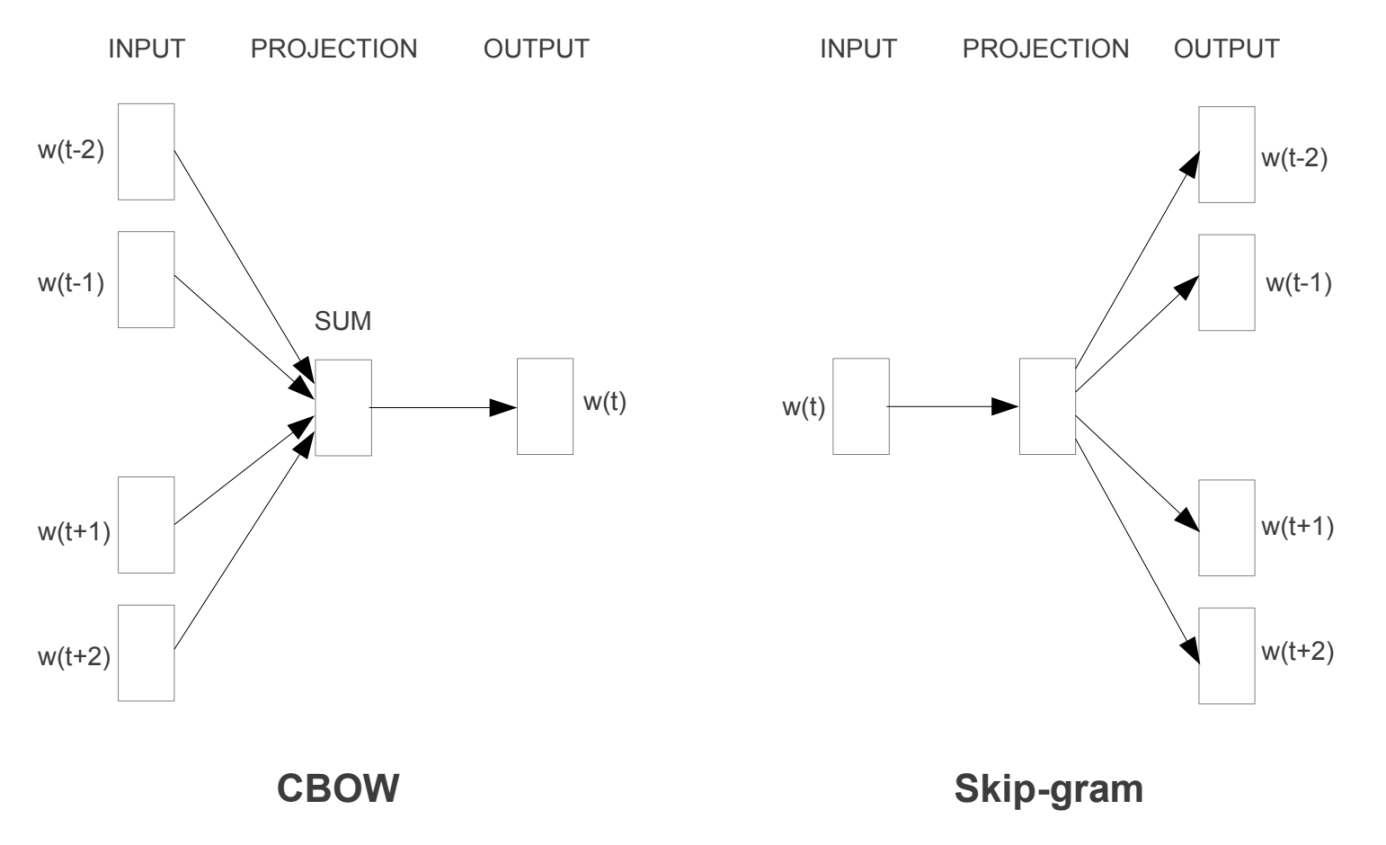
#### Вграждания на думи (Word embeddings)

##### **Описание на алгоритъма**

Вграждане на дума (*word embedding)* [17] наричаме научено представяне на дума или текст, такова че думи със сходно значение имат сходно представяне. Думите са представени като вектори с реални координати в предварително определено линейно пространство. На всяка дума съответства един вектор, като самите вектори се научават по начин, наподобяващ тренирането на невронна мрежа. Размерността на векторите е малка (най-често между 100 и 300) в сравнение с тази на векторните представяния *one-hot encoding* [49]. Тя винаги е много по-малка от броя на всички думи в речника, който моделът използва. Всяко измерение представлява дадена характеристика на думата, но тези характеристики са абстрактни т.е. не са определени явно.

Моделът на вгражданията се основава на така наречената хипотеза за разпределението (*distributional hypothesis*) [18], която гласи че думите, които се използват в сходен контекст, имат сходни значения. Алгоритъмът за трениране на вграждания използва много текстове за обучение, следи контекста в който всяка дума е използвана и думите използвани в сходен контекст получават сходни вектори.

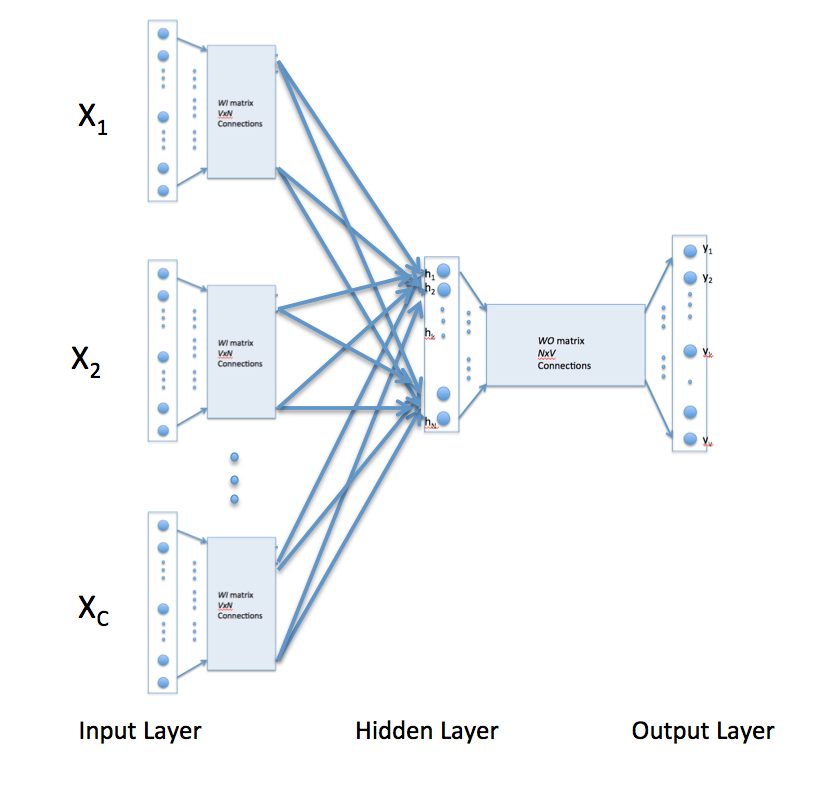
Има множество алгоритми за генериране / изчисляване на вгражданията. Два от най-популярните са ***CBOW (Continuous Bag of words)*** и ***Continuous Skip-gram Model*** [19], чиито архитектури са представени на фиг. 2.



**фиг. 2** *Архитектура на**моделите* ***CBOW(Continuous Bag of words)*** *и* ***Continuous Skip-gram Model****, които се използват за изчисляване на вграждания на думи ( Източник: [19] )*

При обучение ***CBOW*** научава вгражданията, като предсказва дадена дума (наричана още *целева дума*) въз основа на контекста ѝ. Това е целта и на системата за замяна на чуждици и затова по-подробно ще разгледам именно този модел. Нека въведем следните означения:

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



**фиг. 3** *Архитектура на**модела* ***CBOW (Continuous Bag of words)****, който се използва за изчисляване на вграждания на думи [21]*

***CBOW*** работи като невронна мрежа с три слоя – входен, скрит (проекционен) и изходен (фиг. 3) по следния начин [20] [21]:

1. Входният слой има C\*V на брой неврона, които са разделени на C части, всяка от които отговаря за един входен вектор. Средният слой има N неврона, а изходният – V неврона.
2. Входните и изходните вектори на невронната мрежа са с размерност 1xV, като входните са *one-hot encoded* [49].
3. Входните вектори биват усреднени, за да може скритият слой на практика да работи с 1 усреднен входен вектор X.
4. Теглата между входния и скрития слой могат да се представят в матрица WI, наречена входна матрица с размерност VxN.
5. Аналогично теглата между скрития и изходния слой се представят в изходна матрица WO с размерност NxV.
6. Няма активираща функция между слоевете.
7. За всяко множество от входни вектори, невронната мрежа изчислява следното (за улеснение съм посочил и размерността на векторите и матриците):

(7)

1. Грешката се изчислява като разлика на изходната и целевата дума. Използва се обратно разпространение на грешката (*backpropagation*) за обновяване на теглата на мрежата.
2. Към координатите на изходния вектор ***Y*** може да се приложи функцията *softmax* [50]. Тогава i-тата координата на новополучения вектор ***Y’*** е равна на вероятността i-тата дума да е целевата.
3. След завършване на обучението на мрежата, вгражданията се намират в изходната матрица, или по-точно i-тия вектор-стълб от матрицата е вграждане на i-тата дума от речника.

Към стандартния *CBOW* може да се направят няколко промени. *CBOW* с *position-weights* [40] утеглява думите от контекста спрямо това колко са близки те до чуждицата. *CBOW* с *character n-grams* [40] се използва, за да може вграждането на дадена дума да съдържа информация от поддумите ѝ.

Алгоритъмът *skip-gram* се опитва да предскаже контекста на дадена дума спрямо самата дума, което е обратното на това, което прави *CBOW*. Именно затова и архитектурата на *skip-gram* е в обратна на тази на *CBOW*, както е показано на фигура 2.

Тъй като сходните по смисъл думи би трябвало да имат сходни вграждания, друг подход за намиране на най-подходяща заместваща дума в изречение е да се намери тази дума, чието вграждане е най-близко до вграждането на изречението. Вграждането на изречението може да е изчисли като средноаритметичното на вгражданията на всички думи в него или пък да се използва някакво утегляване на думите в изречението.

##### **Вграждания на думите на български език**

*fasttext* [36] имат тренирани вгражданияза множество езици, включително и български. *fastetxt* имат 3 набора от вграждания. Първият ще наричам *CBOW (Word) Vectors*. Той [36] [37] използва алгоритъма *CBOW* с *position-weights* [40] и *character n-grams* [40] с дължина 5 за научаване на 300-измерни вграждания, обучавани върху *Common Crawl* [38] и *Wikipedia* [39]. Вторият модел е *Wiki Word Vectors* [42]. Той използва алгоритъма *skip gram* със стандартни параметри [41]. Векторите са обучени върху текстове от Уикипедия и отново са 300-измерни. Третият модел е *Wiki Aligned Vectors* [43], като целта му е векторите в различните езици да са приравнени помежду си. *Wiki Aligned Vectors* са тренирани върху Уикипедия и след това подравнени с метода *RCSLS* [44].

##### **Предимства и недостатъци на алгоритъма**

Основен недостатък на алгоритъма е, че при научаването на вгражданията не се прави разлика между различните значения на думата. Ако една дума има няколко значения, тогава нейното вграждане най-вероятно ще вектор, който ще е по средата между значенията ѝ (по средата между векторите на сходните думи за всяко значение). Друг недостатък е, че научаването на вгражданията е бавно и изисква много входни данни. Този недостатък се преодолява лесно, ако се използват вече научени вграждания.

Предимство на алгоритъма е, че той е вероятностен, което на теория го прави по-добър от детерминистичните алгоритми [20]. Лесно може да се адаптира към задачата за замяна на чуждица. Вместо да се търси най-добрата целева дума сред всички думи, това търсене може да се направи само измежду синонимите на чуждицата (виж точка 9 от описанието на *CBOW*). Освен това алгоритъмът лесно се справя с думи, които ги няма в речника. Ако за някоя дума от контекста няма вграждане*,* то тя може просто да се изпусне. Същото може да се направи и ако някой от синонимите на чуждицата няма вграждане. Проблем би настъпил само ако нито една дума от контекста или нито една дума от синонимите няма вграждане, което е много малко вероятно.

#### BERT

*BERT* (*Bidirectional Encoder Representations from Transformers*) [22] е езиков модел, който може да се използва за голямо множество от задачи, свързани с обработването на естествен език – задача за предсказване на липсваща дума в текст, отговаряне на въпроси и извличане на изводи от текст и други. Моделът е трениран върху голям брой текстове, които не са анотирани, като самото трениране е двупосочно [22]. Отличителна черта на *BERT* е това, че архитектурата му е почти една и съща за всички задачи, за които може да се използва. Архитектурата на *BERT* многослоен двупосочен кодиращ трансформатор (Transformed encoder) [23]. Трансформаторът използва *sequence to* *sequence* архитектура [24] с механизъм за внимание. *BERT* е трениран върху 2 задачи – задача за предсказване на липсваща дума в текст и задача за предсказване на следващото изречение. Първата задача е точно тази, за която ще използвам модела, като вместо да взема най-вероятната произволна заместваща дума, ще взема най-вероятната измежду синонимите.

Предварително тренираният модел *BERT-Base, Multilingual Cased* [25], който е посочен в оригиналната статия [22] включва 104 езика, сред които и български език. Данните са тренирани върху текстове от Уикипедия, като за тези езици, за които има по-малко страници в Уикипедия, моделът е по-ограничен. Моделът за български език е доста ограничен, тъй като за 2781 (около половината) примера от анотираните данни (които са описани по-подробно в точка 3.3), нито един от синонимите не присъства в модела. Поради това ограничение, ще използвам някой от останалите описани алгоритми при решаването на задачата за замяна на чуждица в текст.

#### Сравнение между алгоритмите

В таблицата, показана по-долу ще сравня разгледаните алгоритми (модели) спрямо няколко критерия:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | N-грами | Вграждания на думи (word embeddings) | Заместване с произволен синоним от синонимен речник | BERT |
| Какъв контекст отчита? | Само последните N - 1 думи преди липсващата | Всички думи в контекста | Никакъв | Всички думи в контекста |
| Търси липсваща дума по-скоро синтактично или семантично | И двете, но по-скоро синтактично | По-скоро семантично | Семантично, защото думите от синонимния речник са близки до значение с дадената | И двете |
| Лесно ли се справя с липсващи / редки за модела думи? | Не | По-скоро да | Не | По-скоро да |
| Лесно ли се адаптира от задачата за предсказване на липсваща дума към задачата за замяна на чуждица? | Да | Да | Да | Да |
| Има ли вече трениран такъв модел за български език? | Да | Да, този на *fastText* | Да, има добри и налични онлайн синонимни речници | Да |
| Съдържа ли достатъчно думи? | Да | Да | Да | Не |

**таблица 1**  *Сравнение на алгоритмите с N-грами, с вграждания на думи, със заместване с произволен синоним и BERT*

### 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) [15] 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. [16] 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 [17] 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.

# Използвани езикови ресурси

За намирането на най-подходящата заместваща дума на дадена чуждица, както и за тестване на системата, са необходими различни езикови ресурси. Именно тях ще опиша в точка 3.

## Синонимен речник

Използването на синонимен речник ще улесни изключително много намирането на най-подходяща заместваща дума. Нека първо разгледаме определението на думата ***синоним***.

***Синоними*** [27] наричаме думи, които имат еднакво или сходно значение в много контексти. От дефиницията следва, че не е задължително два синонима да имат едни и същи значения във всички контексти. На системата за замяна на чуждици се пада отговорността да реши кой от синонимите е най-подходящ спрямо конкретния контекст. Въпреки, че дума и неин синоним не винаги са взаимозаменяеми, синонимите са много добри кандидати за заместващи на чуждици, тъй като имат сходно значение с чуждицата. Друго предимство на използването на синоними е това, че някои алгоритми за намиране на липсваща дума в изречение могат да се сведат много лесно до алгоритми за заместване на дума в текст, като се използват синоними на думата, която ще се замества (това е обяснено по-подробно в точка 2 за всеки отделен алгоритъм).

За намирането на синоними е необходим достатъчно голям синонимен речник. Смятам да използвам синонимния речник на *БГ ОФИС* [15], който съдържа синоними за 22889 думи. За всяка от тези думи е дадено множество от нейните синоними. Това множество е разделено на секции (редове). Секциите са разделени спрямо отделните значения на думата или спрямо отделните нюанси при използването ѝ. Това означава, че синонимите от една (произволна) секция са много по-близки като значение до синонимите от същата секция отколкото до тези от друга. Нека вземем следния пример:

стеснен

(гл.) тесен

(гл.) притеснен, ограничен, затруднен, спънат, умъчнен

(гл.) стеснителен, смутен, неловък

В първата секция има само една дума – „*тесен“*, която се използва най-вече за предмети. Синонимите от втория и третия ред се използват за човек.

Друг добър пример е с думата „*абсолютен“*:

абсолютен

(прил.) положителен, истински, неоспорим, безспорен

(прил.) установен, категоричен, определен, сигурен, точен, изричен, ясен, безусловен, обезателен, формален

(прил.) повелителен, заповеден, заповеднически, рязък

(прил.) неограничен, пълен, самовластен, тираничен, деспотичен, автократичен

(прил.) върховен, независим, суверенен

(прил.) решителен, окончателен

(прил.) отявлен, явен, флагрантен, същински, същи, чист

(прил.) безрезервен

(прил.) цялостен, целокупен

(прил.) самодържавен

(прил.) съвършен, идеален, цял, безукорен, завършен

(прил.) очевиден

Ясно е, че например думите „*върховен“*, „*независим“* и „*суверенен“*, които са от една секция, са доста сходни, но пък и трите са много различни по значение от „*явен“* и **„***положителен“*, които са в съвсем различни секции.

## Тълковен Речник

Тълковен речник е речник, в който са обяснени различните значения на дадена дума, както и различни контексти, в които думата е използвана. Тези значения често са съпътствани с примерни изречения, в които търсената дума е употребена точно в посочения от речника контекст. Именно тази информация ще се окаже много полезна за системата, която разработвам.

Има няколко речника на българския език, които са налични онлайн. За системата съм използвал този, който има достатъчно много думи, има достатъчно много значения и примери за дадена дума и може лесно да бъде свален от интернет и обработен. Спрямо тези критерии избрах *Rechnik Chitanka* [28]. Речникът съдържа около 35 000 думи, като са представени техните значения, част на речта, синоними, възможни грешни изписвания, словоформи и връзки към други речници, които съдържат думата. Тълковната част на *Rechnik Chitanka* е взета от *eurodict.com* [29] и *onlinerechnik.com* [30], а синонимите – от синонимния речник на *БГ ОФИС* [15], който е описан по-подробно в точка 3.1.

Както повечето речници, *Rechnik Chitanka* е добре структуриран. Значението на всяка дума е описано в следния формат:

<словоформи на думата>  
1.<дали значене 1 е преносно или разговорно>   
<обясняване на значение 1; синоними, отговарящи на значение 1>  
<примерни изречения, в която думата е със значение 1>  
2. ...  
...  
n. <дали значене n е преносно или разговорно>   
<обясняване на значение n; синоними, отговарящи на значение n>  
<примерни изречения, в която думата е със значение n>  
<членувани и други форми>

Ако думата има само едно значение, то номерирането на значенията липсва, защото е излишно. Освен това всяка една от частите от формата може да липсва – може да липсват примери за някое значение, да няма синоними за някое значение или някое значение да липсва изцяло. Често се случва контекстовият синоним да е споменат в частта <обясняване на значение n> за дадена дума.

Ето и пример, за това как е описано значението на думата *стерил*ен в речника:

стерѝлна, стерѝлно, мн. стерѝлни, прил.  
**1.** Който е обеззаразен. *Стерилен инструмент.*  
**2.** Който не може да даде потомство, да се възпроизведе; безплоден. *Стерилен мъж.*  
**3.** Прен. Чиито творчески усилия са безплодни. *Стерилен писател.*  
**4.** Прекалено чист; изчистен от всичко, което може да навреди. *Стерилна обстановка.*  
същ. стерѝлност, стерилността̀, ж.

## Анотирани данни

### Изисквания към анотираните данни

Както всяка система, и системата за замяна на чуждици се нуждае от анотирани (тестови) данни, за да провери своята ефективност. Тестовите данни трябва да отговарят на няколко условия.

Анотираните данни се състоят от множество примери. Всеки пример трябва да съдържа три неща:

1. Дума, която искаме да заменим с подходящ синоним в даден текст (целева дума). Тя няма задължително да е чуждица, защото целта на анотираните данни е да провери дали системата намира най-подходящата сходна на целевата дума.
2. Текст / изречение / словосъчетание или по-общо контекст, който съдържа целевата дума
3. Най-добрите заместващи думи за целевата дума. Те могат да представляват списък, подреден от най-подходящата до най-неподходящата замяна или списък само с най-подходящите думи. Желателно е този списък е достатъчно подробен.

### Извличане на анотираните данни

След направено от мен проучване не намерих анотирани данни на български език, които да отговарят на условията, описани в точка 3.3.1. В същото време всеки тълковен речник съдържа изключително много информация за значенията и контекстовите синоними на голямо множество от думи, която може лесно да се извлече от съответния речник. Ето как може да се направи това:

Извличаме следните фрагменти за всяка дума *w*:

1. от тълковния речник се извлича:

за всяко i от 1 до броя на значенията на *w*  
Meaningi = <обясняване на значение i; синоними, отговарящи на значение i>   
Sentencesi = <примерни изречения, в която думата е с значение i>  
  
от синонимния речник се извлича:  
Synonyms = <множество от всички синоними на w от синонимния речник>

1. за всяко i от 1 до броя на значенията на *w*  
   - изчислявамe множеството :

- за всяко изречение *s* от Sentencesi добавяме във файла с анотираните данни  
*w;s;*

(допълнително *s* може да се раздели на множество от думи)

Да видим как ще работи този алгоритъм с думата стерилен:

Meaning1 = {който, е, обеззаразен}  
Sentences1 = {Стерилен инструмент.}  
Meaning2 = {който, не, може, да, даде, потомство, да, се възпроизведе, безплоден}  
Sentences2 = {Стерилен мъж.}  
Synonyms = {стерилизиран, дезинфекциран, обеззаразен, ялов, безплоден}

Изчисляваме :

Като резултат във файла с анотираните данни се записва следното:

стерилен;стерилен,инструмент;обеззаразен  
стерилен;стерилен,мъж;безплоден

Този метод работи, тъй като множеството *Meaning* съдържа контекстовите синоними на дадена дума (в достатъчно много случаи) и към него са дадени примерите в *Sentences*. *Synonyms* се използва именно, за да може да се намерят / филтрират тези синоними измежду всички думи в Meaning.

Намирането на индекса на целевата дума в примерното изречение може да стане като се намери най-сходната дума от примерното изречение с целевата. При търсенето на тази най-сходна дума могат да се направят три неща – да се сравняват стемовете [51] (или основните форми [51] на думите), да се сравняват думите по разстояние на Левенщайн [52] или да се сравняват по най-голям брой съвпадащи букви до първото разминаване при четенето на думата от ляво надясно. И трите метода могат да дадат грешка, ако в изречението има няколко сходни по изписване думи с целевата. Аз съм избрал третия метод, като той даде грешка за един пример. Намирането на индекса на целевата дума се налага само ако се използва утегляване на вгражданията на примерното изречение (утегляванията са описани по-подробно в точка 4.2.3).

Ясно е, че описаният метод за извличане на анотирани данни не е най-идеалният. Един недостатък е, че често *ContextSynonyms* ще е празното множество, защото не винаги *Meaning* ще съдържа контекстови синоними. *Sentences* - също може да е празно в някои случаи. Въпреки това има достатъчно много примери, които могат да се генерират по този начин. Следвайки алгоритъма се получиха 5 397 примера.

Основният недостатък на този подход е, че полученото множество от контекстови синоними не е пълно, тъй като при обясняването на дума в речника, не винаги се изброяват всички възможни синоними. Затова полученото множество ще **наричам ограничено множество от контекстови синоними** или само **ограничено множество (ОМ).**

### Важни статистики за анотираните данни

В този раздел ще разгледам някои важни статистики, отнасящи се до анотираните данни. Първата е разпределение на примерите от анотираните данни по брой синоними от ограниченото множество, показано на фигура 4. От това разпределение се вижда, че повечето примери имат един или два контекстови синонима и че много малък брой примери имат над 3 контекстови синонима. Средният брой синоними от ограниченото множество на пример е само 1,51.

**фиг. 4**  *Разпределение на примерите от анотираните данни по брой синоними от ограниченото множество от синоними*

При разширеното множество от синоними, описано в точка 5.1.1, нещата стоят по друг начин. Очевидно, даден пример има много повече синоними от разширеното множество отколкото от ограниченото – средно 7,22 синонима на пример. Въпреки, че повечето примери имат между 2 и 7 синонима от разширеното множество, се намират и такива, които имат цели 30 и 32 (на фигура 5 са дадени до 30) такива синоними.

**фиг. 5**  *Разпределение на примерите от анотираните данни по брой синоними от разширеното множество от синоними*

Общият брой синоними на пример обаче е много по-висок (фиг.6), като един пример има средно около 21 синонима. 1 191 (22%) примера имат най-много 3 синонима – това означава, че при тях би работил алгоритъмът, който подрежда предложите синоними на случаен принцип, описан в т. 2.3.2. За останалите примери обаче е необходим по-сложен алгоритъм за подреждането на предложените синоними. Забелязва се, че някои думи имат над 90 синонима.

**фиг. 6**  *Разпределение на примерите от анотираните данни по брой синоними от синонимния речник*

Както споменах в точка 3.1, синонимният речник е разделен на секции от синоними спрямо смисъла им. Всеки пример от анотираните данни има средно 5,6 секции. Повечето примери имат до 7 секции, някои обаче имат над 30 (фиг. 7).

**фиг. 7** *Разпределение на примерите от анотираните данни по брой секции*

Повечето изречения от анотираните данни са къси – средната им дължина е само 3,66 думи. Точното разпределение на примерите по брой думи в изречението е дадено на фигура 8.

**фиг. 8**  *Разпределение на изреченията от анотираните данни по брой думи в тях*

Анализът на анотираните данни показа, че не е удачно системата за замяна на чуждици да връща случайни синоними като заместващи думи, защото много от примерите имат голям брой синоними. Освен това поради факта, че изреченията в анотираните данни са къси, не се налага отрязване на думи от тях, които са твърде далеч от целевата дума и които могат да създадат отклонения от цялостния смисъл на изречението.

## Речник на чуждиците в българския език

Речник на чуждите думи може да е полезен за системата, като такъв речник може да има няколко приложения. Първото възниква, когато системата генерира примерни синоними за дадена чуждица. Тогава от този списък може да се премахнат чуждиците, за да не се случи така, че една чуждица да се замени с друга. Това става като се провери дали даден синоним принадлежи на речника на чуждиците.

Друго приложение на речника е, че благодарение на него ще може автоматично да се откриват чуждиците в произволен текст, което би позволило автоматичното им заместване.

Най-важното приложение обаче е, че чуждиците, за които са дадени синоними в речника на чуждиците, могат да се добавят към вече наличния синонимен речник на *БГ ОФИС* [15] с цел обогатяването му. Ето и някои речници на чуждиците, които са достъпни свободно в интернет:

1. Речник на турските думи в български език [11]
2. Речник на редки, остарели и диалектни думи [32]
3. Речник на чуждиците в българския език [33]

## Списък със стопдуми за български език

Стопдумите [26] са най-често използваните думи в конкретен език. Счита се, че те допринасят най-малко за контекста и смисъла на даден текст и затова биват пропускани или премахвани в много от текстообработващите алгоритми (например при търсачките). Речникът на стопдуми за български език [46] съдържа 259 думи и може да е полезен и в задачата за замяна на чуждици.

# Метод за предлагане на замени на чуждици в българския език

## 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*.

# Експерименти за оценка на разработения метод

## Метрики за оценяване на системата

### Ограничено и разширено множество от контекстови синоними

**Ограничено множество от контекстови синоними** за дадена дума *w* и изречение *Sentence* ще наричам множеството от контекстови синоними, които съответстват на *Sentence* в анотираните данни. Него ще бележа още и с *LimitedSet* (очевидно *LimitedSet* *= ContextSynonyms* при фиксирани дума и изречение). Както е описано в точка 3.3.2, това множество не съдържа всички синоними, които са подходящи за контекста на *Sentence*.

За да се реши посоченият проблем въвеждам **разширено множество от контекстови синоними,** което има за цел да добави липсващите контекстови синоними. Добавянето е възможно благодарение на това, че синонимният речник, който използвам [15] за откриването на синоними е разделен на секции (за които стана дума в точка 3.1). Разширеното множество от контекстови синоними за произволна дума *w* се получава по следния начин**:**

1. Нека *SectionSynonyms* е множеството от всички секции, които съдържат синонимите на *w*
2. За всеки синоним от *LimitedSet* намираме секцията от *SectionSynonyms*, в която той се съдържа и добавяме всички думи от тази секция към множеството *ExtendedSynonyms*
3. *ExtendedSynonyms* кръщаваме разширено множество от контекстови синоними за думата *w* в изречение *Sentence*

На пръв поглед разширеното множество от контекстови синоними решава проблема с недостатъчното контекстови синоними при ограниченото. Съществува обаче опасността това разширено множество да съдържа повече синоними отколкото е необходимо. По тази причина, ще използвам и двете множества за оценка на системата.

Смятам, че тъй като и двете множества са получени от речници, те са достатъчно точен показател за това кои синоними са добри и кои не в съответния контекст. Най-точна оценка може да се получи, ако много хора оценят ръчно примерите и отбележат кои синоними са точни и кое не. Това естествено би отнело известно време и със сигурност ще стане по-бавно от автоматичното извличане на множествата.

### Описание на метриките

Както всяка една система, системата за замяна на чуждици с подходящи синоними в текст има нужда от метрики, които да се използват, за да може да се провери ефективността ѝ. Метриките ще трябва да опишат доколко добре системата се справя с целта си – връщане на списък със синоними, в който най-подходящите са възможно най-високо в списъка с *предложенията*.

#### Най-висока позиция на синоним в “предложенията“

Както казахме, най-подходящите заместващи думи трябва да са на възможно най-високи позиции в списъка с *предложенията*. За да се провери дали това е така, въвеждам метриките ***Най-висока позиция на синоним от ограниченото множество в „предложенията“ (НВПСОМП)*** и ***Най-висока позиция на синоним от разширеното множество в „предложенията“ (НВПСРМП)***. За да се изчислят тези метрики, се взимат контекстовите синоними от съответното множество (разширено или ограничено), както и тези, генерирани от системата. След това се взима позицията на най-високо разположения контекстов синоним в предложенията.

За всички примери може да се изчислят *средна**НВПСОМП* (*СНВПСОМП*) и *средна НВПСРМП* (*СНВПСРМП*), която се изчислява като средно аритметично на съответната метрика (*НВПСОМП или НВПСРМП*) за всички примери.

#### Брой / Процент контекстови синоними, които са на първо място в предложенията

Тази метрика ще провери в колко процента от изреченията системата се справя отлично. Под отлично се има предвид такъв случай, в който още първото предложение е контекстов синоним. Изчислява се като броят на отличните примери се раздели на броя на всички примери от анотираните данни.

Аналогично може да се изчисли броят / процентът на примерите, в които контекстовият синоним е в сред първите две, три и т.н. позиции *в предложенията*.

## Постановка на експеримента

Експериментът, който ще проведа е следният: за всеки пример от анотираните данни в два файла ще се запишат съответно получените *предложения*, както и описаните в точка 5.1 метрики. Ще тествам различните разновидности на предложения алгоритъм, описани в точки 4.2 и 4.3 (трите вида разстояния, двата вида утеглявания, трите набора от вграждания, с или без премахване на стопдумите).

За всеки експеримент ще се генерират гореспоменатите 2 файла, които ще се намират съответно в папките *Results* и *Stats* и името им ще съдържа на първо място използвания модел, а след това и неговите специфики, например *wiki\_aligned\_euclidean\_distance\_no\_stopwords*. Фаловете ще имат следната структура:

На първия ред от файла ще е изписано описанието на формата му, а на останалите -самите данни, които ще са в този формат. Форматът е следният:

* За Results –

най-висока позиция на синоним от ограниченото множество в „предложенията“;

най-висока позиция на синоним от разширеното множество в „предложенията“;

целевата дума;

контекстът на целевата думата;

контекстовите синоними на думата от анотираните данни (наречено още ограничено множество от синоними);

разширено множество от синоними на целевата дума;  
списъка с предложенията, като за всяко от тях е дадена съответната му част на речта и разстояние до контекста;  
част от списъка с предложенията, съдържащ само ограниченото множ. от синоними;

част от списъка с предложенията, съдържащ само разширеното множ. от синоними;

* За Stats –

целева дума;

най-висока позиция на синоним от ограниченото множество в „предложенията“;

най-висока позиция на синоним от разширеното множество в **„**предложенията“;  
брой синоними от ограниченото множество от синоними на целевата дума;

брой синоними от разширеното множество от синоними на целевата дума;  
брой на всички синоними на целевата дума от речника;

брой секции от синонимния речник за целевата дума;  
брой думи в контекста на целевата дума;

Целта на *Stats* е да може да се откриват взаимовръзки в данните и да се правят анализи на тях.

За експеримента ще използвам синонимния речник на *БГ ОФИС* [15], защото той съдържа всички целеви думи от анотираните данни. Целта на експеримента, който ще проведа е да се види колко добре системата се справя с намирането на най-подходящ синоним, независимо дали целевата дума чуждица или не. Тъй като за компютъра няма разлика между това дали една дума е чуждица или не, алгоритъмът който предлага заместваща на произволна дума, би трябвало да работи еднакво добре и при заместване на чуждица.

## Анализ на резултатите

### Резултати от алгоритъма намиране на най-близък синоним до контекста без утегляване на думите от контекста

На таблица 2 са дадени резултатите за моделите *Wiki Aligned Vectors* и *CBOW Vectors* при използвани евклидово, косинусово и скаларно разстояние с и без премахване на стопдуми. При провеждането на всички експерименти са изчислени и показани в таблиците метриките *средна най-висока позиция на синоним от ограниченото множество в предложенията* и *средна най-висока позиция на синоним от разширеното множество в предложенията*. В таблицата съм пропуснал резултатите за модела *Wiki Word Vectors*, тъй като те са по-лоши от повечето в таблицата – СНВПСОМП е 6.33 и СНВПСРМП е 2.68 при различните разстояния.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модел | Мярка за разстояние | С или без премахване на стопдуми | Средна най-висока позиция на синоним от ограниченото множество в “предложенията“  (СНВПСОМП) | Средна най-висока позиция на синоним от разширеното множество в “предложенията“  (СНВПСРМП) |
| Wiki aligned vectors | Eвклидово | Без премахване | 5.27 | 2.65 |
| Wiki aligned vectors | Косинусово | Без премахване | 5.27 | 2.65 |
| Wiki aligned vectors | Скаларно произведение | Без премахване | 5.27 | 2.65 |
| CBOW  vectors | Евклидово | Без премахване | 8.30 | 2.99 |
| CBOW  vectors | Косинусово | Без премахване | 5.40 | 2.43 |
| CBOW  vectors | Скаларно произведение | Без премахване | 5.70 | 2.53 |
| Wiki aligned vectors | Eвклидово | С премахване | 5.28 | 2.65 |
| Wiki aligned vectors | Косинусово | С премахване | 5.28 | 2.65 |
| Wiki aligned vectors | Скаларно произведение | С премахване | 5.28 | 2.65 |
| CBOW  vectors | Евклидово | С премахване | 8.23 | 2.93 |
| **CBOW**  **vectors** | **Косинусово** | **С премахване** | **5.30** | **2.39** |
| CBOW  vectors | Скаларно произведение | С премахване | 5.57 | 2.49 |

**таблица 2** *Резултати от алгоритъма най-близък синоним до контекста за Wiki Aligned Vectors и CBOW Vectors при еднакво тегло на всички думи от контекста с експериментиране с три вида разстояния и с експеримент с и без стопдуми*

Забелязва се, че моделът *Wiki Aligned Vectors* не показва почти никакви разлики в стойностите на метриките, независимо от използваното разстояние и премахването или не на стопдуми. За разлика от него, при другия модел *CBOW vectors*, такава разлика има. *CBOW Vectors* се справя най-добре при косинусово разстояние. При скаларно разстояние стойностите на метриките са само малко по-лоши спрямо косинусовото. При евклидово разстояние резултатите са най-лоши, като достигат цели 8.30 за *СНВПСОМП*. При премахване на стопдуми, резултатите при модела *CBOW Vectors* са по-добри.

Най-добрият резултат за *Wiki Aligned Vectors* е 5.27 за *СНВПСОМП* и 2.65 за *СНВПСРМП*. За *Wiki Aligned Vectors* най-добър е резултатът при косинусово разстояние без стопдуми, а именно 5.30 за *СНВПСОМП* и 2.39 при *СНВПСРМП* (отбелязани в зелено на таблица 2).

При следващите експерименти ще се използва косинусово разстояние и премахване на стопдуми, освен ако не е посочено друго, защото те дават най-добри резултати.

### Резултати от алгоритъма намиране на най-близък синоним до контекста с утегляване на думите от контекста

Следващият експеримент (таблица 3) е с линейно утегляване на думите от контекста за модела *CBOW Vectors*. Резултатите от него подобряват тези, при които не се използва утегляване, като достигат 2.41 (СНВПСРМП) и 5.14 (СНВПСОМП) при косинусово разстояние. При *Wiki Aligned Vectors* не се наблюдава разлика при утегляване. Възможно е векторите в този модел да са на по-големи разстояния едни от други и затова теглата да не оказват голямо влияние.

|  |  |  |  |
| --- | --- | --- | --- |
| Модел | Мярка за разстояние | Средна най-висока позиция на синоним от ограниченото множество в “предложенията“  (СНВПСОМП) | Средна най-висока позиция на синоним от разширеното множество в “предложенията“ (СНВПСРМП) |
| CBOW  vectors | Eвклидово | 7.75 | 2.90 |
| CBOW  vectors | **Косинусово** | 5.14 | 2.41 |
| CBOW  vectors | Скаларно произведение | 5.64 | 2.57 |

**таблица 3** *Резултати от алгоритъма най-близък синоним до контекста за CBOW Vectors при линейно утегляванене всички думи от контекста, без стопдуми и експериментиране с три вида разстояния*

Таблица 4 представя различни утеглявания, като разликите в резултатите при тях не са особено големи. Най-добър резултат се постига при тегло „*2x + 1“* – 5.13 (СНВПСОМП) и 2.41 (СНВПСРМП).

|  |  |  |  |
| --- | --- | --- | --- |
| Модел | Утегляване | Средна най-висока позиция на синоним от ограниченото множество в “предложенията“ (СНВПСОМП) | Средна най-висока позиция на синоним от разширеното множество в “предложенията“ (СНВПСРМП) |
| CBOW vectors | 2x | 5.14 | 2.41 |
| CBOW vectors | 2x -0.5 | 5.15 | 2.43 |
| CBOW vectors | 2x + 1 | 5.13 | 2.41 |
| CBOW vectors | 2x + 2 | 5.13 | 2.41 |
| CBOW vectors | x2 | 5.20 | 2.47 |
| CBOW vectors | x3 | 5.30 | 2.52 |

**таблица 4** *Резултати от алгоритъма най-близък синоним до контекста за CBOW Vectors при различни утеглявания и използвано косинусово разстояние без стопдуми*

### Резултати от алгоритъма, който използва невронната мрежа на CBOW

Таблица 5показва резултатите от използването на научената от алгоритъма CBOW невронна мрежа. Използвал съм скаларно разстояние (както е по алгоритъм). Пробвал съм и косинусово, както и най-успешното утегляване от предишния алгоритъм. Резултатите са добри, но не подобряват най-добрите, постигнати досега.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модел | Мярка за разстояние | Утегляване | Средна най-висока позиция на синоним от ограниченото множество в “предложенията“ (СНВПСОМП) | Средна най-висока позиция на синоним от разширеното множество в “предложенията“ (СНВПСРМП) |
| CBOW  vectors | Скаларно | без | 2.47 | 5.37 |
| CBOW  Vectors | Скаларно | 2х + 1 | 2.53 | 5.44 |
| CBOW  Vectors | Косинусово | 2х + 1 | 2.43 | 5.18 |

**таблица 5** *Резултати при използване на невронната мрежа на CBOW при различни утеглявания и разстояния*

### Анализ на най-добрите резултати от приложените алгоритми

В този раздел ще разгледам по-подробно най-добрите резултати от приложените алгоритми, а те са:

* при *Wiki Aligned Vectors* и най-близка дума до контекста всички резултати са почти еднакви, аз съм взел евклидово разстояние, без тегла и със стопдуми
* при *CBOW* Vectors и най-близка дума до контекста съм взел косинусово разстояние без стопдуми, с тегло 2х + 1
* при използване на невронната мрежа на *CBOW* съм взел алгоритъма с косинусово разстояние без стопдуми, с тегло 2х + 1

На фигури 11 и 12 съм показал сравнение между резултатите от гореизброените три алгоритъма. Двете фигури показват броя примери спрямо най-високата позиция на синоним в предложенията при ограничено и разширено множество. От фигури 11 и 12 се вижда, че няма съществени разлики в този показател при трите алгоритъма. Системата поставя правилния синоним от ограниченото множество още на първо място при около 1400 примера при трите алгоритъма и 3050-3250 при разширеното множество (което е над 50% от примерите).

**фиг.11** *Брой примери спрямо най-високата позиция на синоним от ограниченото множество в предложенията*

**фиг.12** *Брой примери спрямо най-високата позиция на синоним от разширеното множество в предложенията*

По-подробно разбиване на позициите на синонимите съм дал за алгоритъма, който се справя най-добре по метриките *СНВПСОМП* и *СНВПСРМП* (най-добре се справя алгоритъмът най-близък синоним до контекста с CBOW Vectors, косинусово разстояние и тегло 2х + 1) (фиг. 13 и 14).

**фиг.13** *Позиции на синонимите от ограниченото множество в предложенията от алгоритъма най-близък синоним до контекста с CBOW Vectors, косинусово разстояние и тегло 2х + 1*

Синоним от ограниченото множество е на следните позиции в предложенията при:

* първо място – 1406 примера (26.06 %)
* от първо до трето място – 2956 примера (54.78 %)
* от първо до пето място – 3736 примера (69,23 %)

**фиг.14** *Позиции на синонимите от разширеното множество в предложенията от алгоритъма най-близък синоним до контекста със CBOW Word Vectors, косинусово разстояние и тегло 2х + 1*

Подобна статистика за брой примери спрямо позициите на синонимите има и за разширеното множество:

* първо място – 3228 примера (59.82 %)
* от първо до трето място – 4519 примера (83.74 %)
* от първо до пето място – 4917 примера (91.12 %)

Данните показват, че системата за замяна на чуждици се справя със зададената ѝ цел, като успява да намери правилния синоним в голям процент от случаите.

### Анализ на причините, които влияят на успеваемостта на системата

Ще разгледам някои от примерите, в които системата не се справя добре с подреждането на предложенията при алгоритъма с най-добри резултати (най-близък синоним до контекста при *CBOW Word Vectors*, косинусово разстояние и тегло *2х + 1*) . При преглед на някои от тези примери се забелязва, че системата има предпочитание към заместващи думи, които имат същия корен, като този на целевата дума:

Целева дума: мрачен  
Изречение: Мрачна епоха.  
Предложения: ***сумрачен***, навъсен, неприветлив, печален, безрадостен  
НВПСРМП:66, НВПСОМП:66

Целева дума: стар  
Изречение: Всичките ми дрехи са стари.  
Предложения: ***възстар, старик, старичък, старомоден, старинен***   
НВПСРМП:38, НВПСОМП:38  
Разширено множество от синоними: употребяван, използван

Целева дума: душа  
Изречение: Душата на компанията.  
Предложения: ***душевност***, мисъл, природа, сърце, съвест  
НВПСРМП:34, НВПСОМП:34  
Разширено множество от синоними: вдъхновител, организатор, ръководител, център

Други думи, при които системата предпочита заместващи със същия корен са:

слаб – възслаб, слабичък (Изречение: Слабо момченце.)  
пълен – препълнен (Изречение: Пълни крака.)

Друг анализ на резултатите съм представил на таблица 6.

|  |  |  |
| --- | --- | --- |
|  | СНВПСОМП | СНВПСРМП |
| Брой синоними от ОМ | -0,14 | -0,06 |
| Брой синоними от РМ | 0,03 | -0,22 |
| Брой синоними в синонимния речник | 0,41 | 0,36 |
| Брой секции в синонимния речник | 0,35 | 0,35 |
| Брой думи в изречението | 0,02 | 0,00 |
| % на синонимите от ОМ спрямо всички синонимите | -0,31 | -0,21 |
| % на синонимите от РМ спрямо всички синонимите | -0,30 | -0,39 |

**таблица 6** *Корелации в резултатите от алгоритъма най-близък синоним до контекста при CBOW Vectors, косинусово разстояние и тегло 2х + 1*

Таблица 6 представя корелациите на резултатите от алгоритъма най-близък синоним до контекста при *CBOW Vectors*, косинусово разстояние и тегло *2х + 1*. Изследвани са корелациите между метриките (СНВПСОМП, СНВПСРМП) и следните показатели за всеки пример от резултатите – брой синоними от ОМ, брой синоними от РМ, брой синоними в синонимния речник, брой секции в синонимния речник, брой думи в изречението, процент на синонимите от ОМ спрямо всички синоними и процент на синонимите от РМ спрямо всички синоними. От данните могат да се изведат следните изводи:

* Няма корелация между броя думи в изречението и СНВПСРМП (отбелязани в жълто на таблица 6). Това може би се дължи на факта, че повечето изречения в примерите имат дължина 3-5 думи и малко изречения се отклоняват от тази дължина.
* СНВПСОМП и СНВПСРМП имат много ниска корелация с броя думи от ОМ и РМ (между 0 и -0,22), което предполага че броят синоними в тези множества не е решаващ фактор за успеха на системата
* Има средно голяма корелация (между 0,35 и 0,41 по абсолютна стойност) между двете метрики (СНВПСОМП и СНВПСРМП) и всеки от трите показателя – брой синоними от синонимния речник, брой секции от синонимния речник и процент на синонимите от РМ спрямо всички синоними (отбелязани в зелено на таблица 6).

Както споменах в точка 5.1.1, СНВПСРМП е по-добра метрика от СНВПСОМП, а тя корелира най-много с процента на синонимите от РМ спрямо всички синоними. (корелацията е -0,39). Хипотезата, която ще разгледам е следната - системата не се справя добрe с примери, в които целевата дума има много синоними, а разширеното множество съдържа малко синоними. За да проверя тази хипотеза, за всеки пример изчислявам процента на броя синоними от разширеното множество в множеството от всички синоними (които ще наричам процент на синонимите от РМ). След това разпределям примерите спрямо НВПСРМП в групи от по 5. За всяка група изчислявам средноаритметичня процент на синонимите от РМ и получавам фиг.15.

**фиг.15** *Процент на синонимите от разширеното множество спрямо НВПСРМП*

От фигура 15 става ясно, че по-големият процент синоними от РМ влияе на успеваемостта на системата. Това влияние обаче намалява за тези примери, за които НВПСРМП е над 16. Това означава, че процентът синоними от РМ не е единственият фактор, който влияе на системата, т.е. изказаната хипотеза за връзката между двете (НВПСРМП и процент синоними в РМ) е вярна само частично.

Корелацията мeжду НВПСРМП и броя секции в синонимния речник (0,35) предполага, че многото секции в синонимния речник отговарят на голямо разнообразие от значения на целевата дума, което затруднява разпознаването на точното значение. За същото свидетелства и корелацията между НВПСРМП и броя синоними, която е 0,36. Тези връзки обаче не са толкова силни, защото корелация от 0,35 или 0,36 в статистиката се счита за умерена.

Сред другите фактори, влияещи на успаваемостта на системата, със сигурност е посоченото наблюдение, че системата предпочита думи със сходен корен, като този на целевата, както и това, че в някои случаи целевите думи са употребени в по-метафоричен контекст (като например - ***бедно*** въображение, ***пламенен*** поглед, ***набивам*** се в очите, ***пуста*** душа, ***движи*** го омразата, колко пари на баща си е ***изял, здрав*** съвет).

# 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.

# Библиография

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

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

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

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

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

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

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

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

[21] 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>