

**Софийски университет ,,Св. Кл. Охридски”**

Факултет по математика и информатика

**Курсов проект**

**Класифициране на груби коментари към статии в интернет**

Извличане на информация и

Подходи за обработка на естествен език

Студент: Николай Димов Паев

Факултетен номер: 2MI3400435

Магистърска програма: Изкуствен Интелект

Преподаватели: проф. Иван Койчев, проф. Галя Ангелова

Декларация за липса плагиатство:

– Плагиатство е да използваш, идеи, мнение или работа на друг, като претендираш, че са твои. Това е форма на преписване.

– Тази курсова работа е моя, като всички изречения, илюстрации и програми от други хора са изрично цитирани.

– Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.

– Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.

05.02.2024 г. Подпис на студента:

**Съдържание:**

**I Увод**………………………………………………………………………………………………….**2**

**II Първа част (Извличане на информация)**

1. **Въведение**…………………………………………………………………………………..**3**
2. **Данни**…………………………………………………………………………………………**3**
3. **Създаване на речника за поправки**..…………………………………………………**3**
4. **Обработка на коментарите**…….......………………………...…………………………**4**
5. **Коригиране на правописа**………………………………………………….……………**4**
6. **Създаване на речника от груби думи:**…………………………………….…………**5**
7. **Класификация**……………………………………………………………………………...**6**
8. **Експерименти**………………………………………………………………………………**6**
9. **Класификация след корекция със SymSpell**…...……………………………..……**8**
10. **Заключение**…………………………………………………………………………………**9**
11. **Бъдеща работа**………………………………………………………………………...…**..9**

**III Втора част (Подходи за обработка на естествен език)**……………………………....**10**

1. **Въведение**…………………………………………………………………………………**10**
2. **Резултати от предишна работа**………………………………………………….…...**10**
3. **Данни**……………………………………………………………………………………..…**10**
4. **Обработка на коментарите**…………………………………………………………..........................…**10**
5. **Векторизация**…………….……………………………………………………………….**11**
   1. **Токенизация с претрениран токенайзер и векторизация**……………..**11**
   2. **Векторизация след корекция на правописа**……………………………...**11**
   3. **Устойчиво на шум влагане**……………………………………………………**11**
   4. **Векторизация с Fast Text**…………………………………….....................................…………………**13**
6. **Класификация**…………………………………………………………….………………**14**
7. **Конволюционен класификатор**………………………………...……………**14**
8. **Рекурентна невронна мрежа**…………………………………………………**15**
9. **Заключение**……………………………………………………………………..…………**16**
10. **Бъдеща работа**………………………………………………………………………...…**16**

**IV Използвана литература и източници**………………………………………………….…**17**

**V Връзка към хранилище**……………...…………………………………………………….…**17**

**I Увод:**

С разрастването на броя потребители в Интернет става все по-невъзможно ръчното филтриране на грубите коментари по форуми. За целта е удобно да се използват инструменти, които да маркират мръсните коментари и директно да ги премахват от форумите или да ги предоставят на администратор за разглеждане. В първата част на проекта ще бъдат разгледани методи за откриване на грубости на базата на близост на думи. Във втората част ще бъдат засегнати методи за класифициране чрез машинно самообучение.

**Първа част (Извличане на информация)**

**Въведение:**

Коментарите, които се пишат в интернет са обект на множество правописни и други грешки. За да се класифицират надеждно е необходимо да се приведат в по-нормален вид:

* текстът да се токенизира
* сгрешените думи да се поправят
* текст, който е написан на латиница да се кирилизира
* да се изчистят от небуквени символи

Следователно ще ни е необходим речник с правилно изписани форми. За самата класификация ще бъде необходим също и речник с мръсни думи. Подходът който ще бъде имплементиран е да се коригират коментарите и след това да се класифицират на базата на наличие на мръсна дума в тях. Ще се разгледат различни параметри за корекцията, както и различни големини на речника с мръсни думи. За корекцията ще се използва Разстояние по Левенщайн и Жакардов коефициент.

Ще бъде разгледано и съществуващо решение за корекция на правопис: SymSpell, чиито автори обявяват за значително подобрение в ефективността за корекция спрямо разстоянието на Левенщайн.

**Данни:**

Данните, с които ще се работи са коментари извлечени от статии в публицистичния сайт Блиц.БГ. Ръчно анотирани са около 2000. Те ще служат за тестово множество върху, което ще бъдат направени експерименти. Анотираните коментари са класифицирани на базата на смисъл, а не непременно на наличието на мръсни думи в тях. Затова не могат да се очакват много високи резултати от разглеждания вид класификация.

За подготвяне на речника за поправки са извлечени статии от Блиц.бг, Пик.бг и Днес.бг. Използван е същия жанр като коментарите. Предполага се, че те са минали одобрение от редактор и са сравнително правилно написани. (Това предположение е грешно, но това не ни интересува, тъй като задачата е класификация, а на корекция на правопис. Корекцията тук е по-скоро начин за унифициране на думите от коментарите).

Извлечен е също и популярния онлайн речник БГЖаргон, където са поместени 21131 жаргонни думи и изрази с описание, примери и близки думи.

Намерен е и списък от груби думи, които са получени след автоматичен превод на съществуващи списъци на руски, полски и английски. След превода, думите са прегледани ръчно. Тяхната бройка е 100.

**Създаване на речника за поправки:**

Речникът се създава от заглавията и от тялото на статиите. Токенизира се по интервали, препинателни знаци и тирета.

Премахнати са еднобуквените токени, които не са думи, дубликатите, както и думите, които се срещнати по-рядко от 2 пъти. След това са подредени по честота. Финалния им брой е 112 987.

Към този речник добавяме и записите в БГЖаргон, като предварително се токенизират, както и началния списък от груби думи.

**Обработка на коментарите:**

Преди да пристъпим към класификация е необходимо да се обработят коментарите. Първата стъпка е кирилизацията - транслитериране на латински думи чрез директно заместване по правила от вида:

* sht -> щ
* sh -> ш
* 6 -> ш

Правилата са сортирани обратно лексикографски по антецедента и заместването става като се опитат да се приложат в този ред. Така ще се замества като първо се пробва най-дългата възможна поредица.

След транслитерирането се пристъпва към токенизацията. Като за начало е опитано да се токенизира по интервали и знаци. Това обаче не е оптимално решение, тъй като наблюдаваме коментари във вида “м н о г о” или “тъ\*ак”. В този случай ще се получат множество малки токени и ще изгубим възможност за класифициране на думата. Вариант е да се премахват всички символи различни от буква, цифра или интервал. Това решение също води до проблеми, тъй като, ако интервалът е изпуснат, ще се слепят последна и първа дума от съседни изречения. Сегментирането на изреченията пък е задача, която е не по-лесна. Затова е спряно до тук, а експерименти с токенайзъра ще се направят по-късно при оценката на класификацията.

**Коригиране на правописа:**

Токените ще бъдат токенизирани като се използва редакционната им близост спрямо токените от речника за корекции. Възможно е една дума да е еднакво близка с повече от една друга - в такъв случай се позволява предлагането на повече от една дума.

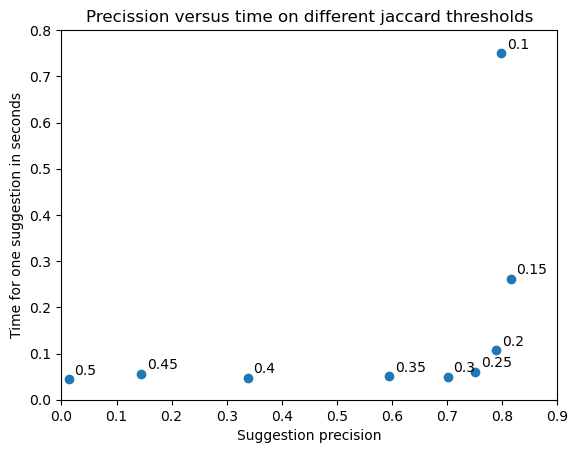
Традиционно за редакционна близост се ползва разстоянието по Левенщайн-Дамерау [1][2]. То обаче е изчислително тежко и сравняването на думата с всяка друга от речника ще е бавно. За целта се намаляват кандидатите за сравнение като се ограничават до тези, които имат достатъчен брой близки биграми [3] с текущия токен. За оценката на близостта е ползван Жакардов коефициент [4]. Откриването на близките думи става чрез индексиране по биграмите. Добавянето на индекс значително повишава ефективността.

Достойни за кандидати са тези думи, за които Жакардовата близост на множеството им биграми с тези на токена е над определен праг. Този праг ще е параметър, който ще се научи експериментално. След това кандидатите ще бъдат сравнявани по Левенщайн.

**Трениране на Жакардовия праг:**

Ще бъде опитано да се намери праг близък до оптималния по отношение на скорост на търсене и точност. Този праг ще зависи от големината на речника. Точността ще се измерва спрямо резултатът, който би върнал разстоянието по Левенщайн. С други думи ще се намери компромисен вариант за подобряване на скоростта на разстоянието по Левенщайн без да се намалява точността.

Експериментите ще бъдат направени върху произволна извадка от 100 думи от речника, който са зашумени в 10 различни кандидати всеки с точно 2 случайни редакции от вида: добавена/ променена/ изтрита буква или разменени 2 съседни букви. Ще бъде тествано каква е точността и скоростта на поправка към оригиналната дума. Резултатите са изложени в графиката.



Графика 1: Точност към време за корекция с различни жакардови прагове

От графика 1 се вижда, че добър компромис за тази големина на речника се получава при праг около 0.25.

**Създаване на речника от груби думи:**

В БГЖаргон думите имат тагове - близки други думи. Идея за разширяване на началния речник с груби думи е да бъдат добавени към него и думите от БГЖаргон, които имат тагове - груби думи. Съпоставянето може да се направи директно, но и чрез разстояние по Левенщайн < 2. След това се филтрират тези записи в БГЖаргон, които не са самостоятелни думи и се получават 2590 груби думи. Повтаряне на този процес с ново добавените думи в списъка с груби думи води до разширяване на речника до около 3469 думи.

Така са създадени поредица от речници с нарастващ размер:

* Речник 1 - с оригиналното множество от мръсни думи
* Речник 2 - тези думи от БГ Жаргон, които имат тагове в първоначалния речник
* Речник 3 - тези думи от БГ Жаргон, които имат тагове, близки до първоначалния речник
* Речник 4 - тези думи от БГ Жаргон, които имат тагове в речник 3

**Филтриране на мръсните думи:**

В процеса е възможно да попадне дума, която не е мръсна в речника от мръсни думи. За целта ще бъдат филтрирани думите, които присъстват в речника за поправки, изваден от статиите.

Експериментално ще бъде проверено, каква големина на речника е оптимална, тъй като прекаленото му разширение, може да доведе до намаляване на прецизността.

**Класификация:**

Идеята за класификация е да бъде отбелязван коментар като груб, при наличието на токен в него, който не е по-близък до не мръсна дума, отколкото до мръсна. С други думи, това са токените, за които списъка от най-близки корекции съдържа мръсна дума.

Тук възможен параметър за оптимизиране е да се допуска и не чак толкова близки думи в списъка от възможни корекции - да има наличие на толеранс. Експериментално ще се провери, кой е по-добрия вариант.

**Експерименти:**

Проведени са експерименти върху анотирания корпус, като са взети равен брой положителни и отрицателни примери. За класифициране е използван речник с мръсни думи с различен размер.

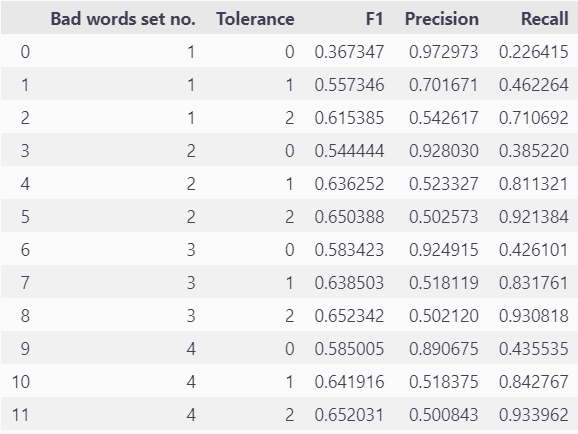


Таблица 1: Резултати от експерименти с различни речници по обем

речници за мръсни думи и различен толеранс.

Увеличаването на речника и повишаването на толеранса (Таблица 1) води до намаляване на прецизността. При голям речник и толеранс от 3, се получава много висок обхват и 50% прецизност, което означава, че алгоритъма класифицира почти всичко в класа на грубите коментари.

При малък речник (Речник 1) и нулев толеранс се наблюдава много висока прецизност от 97%, но и нисък обхват - само 21%.

Обхватът расте с увеличаване на речника, но прецизността пада леко.

Резултатите от експерименти върху анотирания корпус показват, че по-високия толеранс намалява драстично прецизността, така че тази идея ще бъде загърбена.

Важно е да се припомни, че не може да се очаква голям обхват, тъй като коментарите са анотирани на базата на семантиката им, а не на непременното наличие на мръсни думи. По-добри резултати се наблюдават при речник 3 и 0 толеранс - 92% прецизност и 42% обхват.

Идея за увеличаване на обхвата е да се ползва по-агресивна схема за предварителна обработка - премахват се всички не буквени символи и след това се токенизира по интервали. Тази схема за обработка не е оптимална, тъй като ще бъдат слети токени, разделени само от знак за пунктуация, а не интервал. Например “Здравейте,господа”. Предполага се, че това няма да е голям проблем, защото рядко хората пишат така.

Коментарите често съдържат думи от вида “тъ…панар”. Такава схема би се справила в токенизирането на тази дума като един токен.

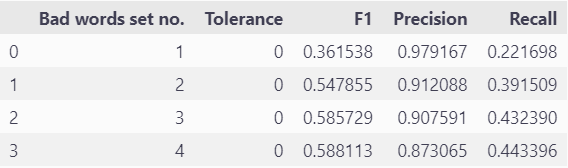


Таблица 2: Резултати от експерименти с различни речници по обем

речници за мръсни думи при агресивна предварителна обработка.

Наблюдава се леко увеличение на обхвата за сметка на прецизността, но общо не води до подобрение. Тази идея също не води до по-добри резултати.

**Къде греши метода:**

Класа на фалшиво позитивните е съставен най-вече от коментари, съдържащи редки и жаргонни думи, които не са мръсни, но нямат аналог в речника за поправки и са близки до някоя дума от списъка с мръсни. Решението на този проблем е създаването на по-добър речник, което е трудоемка задача.

Класа на фалшиво отрицателните се състои от коментари, които съдържат думи, които не се намират в списъка от мръсни, но са употребени в обиден смисъл. Изисква се разпознаване на значението, което е трудна задача.

Класификацията би се подобрила, ако разполагахме с пълен и точен речник на мръсните думи с всичките им форми.

**Класификация след корекция със SymSpell:**

Известно решение за корекция на правопис е библиотеката SymSpell [5]. Тя коригира по близост спрямо разстоянието на Левенщайн, но смята само броя на изтривания, необходими да се достигне една дума до друга. Библиотеката създава модел на речника за поправки, като по този начин извършва част от работата предварително и това прави поправката на една дума значително по-бърза. Удобна функция е, че е способна да коригира поредица от думи, като отделя дори залепени думи.

Недостатък е, че когато се ползва с тази своя функция, библиотеката не предлага повече от едно предложение. Това предложение по-принцип е най-вероятното спрямо честота на думите. В настоящия случай, тъй като речника за поправка е получен от различни източници, не е възможно да се изчисли честота на терм.

Параметър, който подлежи на трениране е максимално разстояние. То се задава

предварително при инициализация. Възможно е да се получи подобрение при добавяне на по-голяма тежест за поправка към мръсна дума, това става с добавяне на по-висока честота на тези думи.

Експериментира се с различни стойности на този параметър върху речник 3 от мръсни думи. (Той даде малко по-добри резултати в предишните експерименти)

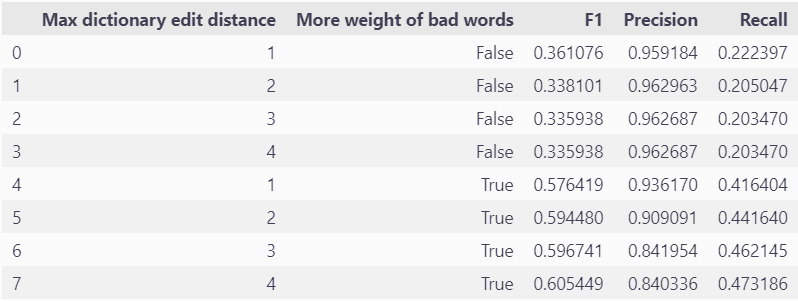


Таблица 3: Резултати от експерименти при използване на SymSpell

за корекция на коментари

Резултатите са представени в Таблица 3. Вижда се, че внасянето на по-голяма тежест за корекция към мръсни думи повишава обхвата повече от двойно. Друго наблюдение е, че разстояние за поправка от 4, не дава по-добри резултати от разстояние 3.



Таблица 4: Резултати от експерименти при използване на SymSpell

и различни речници от мръсни думи

Използването на по-големия речник номер 4 не води до по-добри резултати - прецизността намалява, а обхватът не расте. Най-добри резултати получаваме за речник 3 и максимално редакционно разстояние от 2.

Резултатите при този метод не се различават много от предния хибриден метод за корекция.

**Заключение:**

В настоящия проект бяха разгледани методи за корекция на правопис от друга светлина - бяха използвани за класификация на коментари на базата на наличие на мръсни думи в тях. Предложен беше и начин за автоматично създаване на речник от мръсни думи. Експериментите показаха, че резултатите пряко зависят качеството на използвания речник. По-големия му обем, обаче не е по-добър избор, а напротив - води до намаляване на прецизността, тъй като наличието на много думи в него прави по-голям шанса за случайно съвпадение на неизвестна дума.

Очаквано не беше постигнат висок обхват, но това е ограничение на разглеждания подход за класификация, тъй като той се основава само на близост на думи, а не на семантика. Въпреки това, тук беше дадена добра основа, върху която да се надгражда с по-софистицирани методи.

Лесно разширение ще е да се посочва и конкретната мръсна дума, тъй като разгледаните тук алгоритми сравняват дума по дума.

**Бъдеща работа:**

Целта на настоящия проект беше да се класифицират коментари на базата на наличието на мръсни думи в текста. Възможно разширение би било да се класифицират коментари като мръсни, ако смисъла им е мръсен, дори и без да има мръсни думи в него. За тази цел биха били необходими повече анотирани данни, а за класификацията може да се използва алгоритъм за машинно самообучение.

**Втора част (Подходи за обработка на естествен език)**

**Въведение:**

Традиционно за класифициране на текстове се използват субсимволни методи, които работят с влагания на думи във векторно пространство. Начините за създаване на векторното представяне използват предварително създаден речник. Това представяне не е подходящо при класифицирането на коментари, тъй като те са в лошо качество - съдържат множество жаргонни думи и правописни грешки. Голямо количество думи ще са невиждани до момента. Това ще попречи на класификацията. Известно решение на този проблем е токенизирането на поддуми. Ще бъдат направени експерименти с такова, както и с друго влагане, което е устойчиво на шум, а накрая и с влагане с алгоритъма Fast text. В настоящия проект, ще бъдат проведени и експерименти с различни алгоритми за класификация.

**Резултати от предишна работа:**

В първата част на проекта бяха разгледани методи за класифициране на базата на близостта на думите в коментара до автоматично генериран речник от мръсни думи. Резултатите могат да се обобщят с нисък обхват (recall) около 50% и по-висока прецизност - около 90%. Те не са изненадващи, тъй като такива методи разглеждат думите самостоятелно, а често те придобиват мръсен смисъл, когато са във фраза. Текущия проект има за цел да атакува този проблем.

**Данни:**

Данните, с които ще се работи са коментари извлечени от статии в публицистичните сайтове Блиц.БГ и Днес.БГ. Заедно наброяват близо 120 000. Предвид трудоемката задача по ръчната анотация, анотирани са само 4000 коментара. Те са класифицирани на базата на смисъл, а не непременно на наличието на мръсни думи в тях. Тези от тях, които са получили положителна класификация (съдържат нецензурен език) имат и пример - извадка от коментара, която има мръсен смисъл. Съотношението на класовете мръсни към чисти коментари е 4.5 : 10.

Извлечен е също и популярния онлайн речник БГЖаргон, където са поместени 21131 жаргонни думи и изрази с описание, примери и близки думи.

**Обработка на коментарите:**

Коментарите са в неудобен вид за каквато и да е класификация, някои от тях са със слепени думи, а други са на латиница. Приложена е евристична схема за кирилизирането им. Тя е по-подробно разгледана в предния проект по Извличане на информация. Следващата им обработка е разделянето на думи. Разделяме по интервали и по препинателни знаци. Това не е оптимална схема, тъй като в коментарите се наблюдават често думи от вида: ,,тъ…..ри”, които неправилно биха били разделени в различни думи.

За справяне с зашумените думи в коментарите са сравнени 4 подхода:

* токенизация на под думи и латентен семантичен анализ
* токенизация на под думи след корекция на правописа
* експериментална векторизация, устойчива на шум
* векторизация с Fast text

**Токенизация с претрениран токенайзер и векторизация:**

Приложен е автоматичния алгоритъм за токенизация: Wordpiece [6]. Той е обучен върху некласифицираните данни. Той работи като разделя всички думи първоначално на токени от един символ, а след това на всяка итерация слива най-често срещаните съседни токени в рамките на една дума. Тези операции по сливане се извършват до достигането на предварително определен размер на речника. Избран е размер от 30000, който често се ползва.

Примерни разбивания с тренирания токенайзер:

,,Кво става бе животно" -> 'кво', 'става', 'бе', 'животно'

,,детето ми е първолаче” -> 'детето', 'ми', 'е', 'първола', 'че'

,,непротивоконституционствувателствувайте" -> 'непро', 'ти', 'вок', 'он', 'ститу', 'цион', 'ству', 'вате', 'лс', 'тв', 'ува', 'йте'

Токенизацията на поддуми решава в известен смисъл проблема с непознатите до момента думи - те няма да са неизвестен тоукън, но ще бъдат разбити на незначителни от семантична гледна точка единици. Освен това проблемни са правописните грешки:

,,тъпак” -> 'тъпак'

,,тъоак” -> 'тъ', 'о', 'ак'

Векторизацията след разбиване на токени става лесно, тъй като вече има фиксиран речник от 30 000 думи. Използвана е техниката за латентен семантичен анализ и е научено контекстно влагане от неанотираните данни. [7] След това е приложен Singular Value Decomposition за намаляване на размерността. Използвана е имплементацията на SVD от библиотеката Scikit Learn [8].

**Векторизация след корекция на правописа**

Качеството на текста в коментарите е твърде ниско и алгоритмите за машинно самообучение не биха се справили добре с тях. Ще бъде разлгедана и идеята за предварително коригиране на коментарите спрямо речник от думи, извадени от статиите, както и от БГЖаргон. След това думите ще се токенизират и векторизацията ще бъде извършена подобно на тази с поддуми.

**Устойчиво на шум влагане:**

Сред коментарите много често потребителите правят ,,цензуриране” като скрият някоя буква от нецензурна дума с някои символ. Бихме искали алгоритъма да не се влияе от такова скриване.

Би било хубаво, ако имахме начин да получим влагане, при което близки по букви токени се представят и като близки вектори. В общия случай това е трудна задача, тъй като функциите за близост на стрингове от типа на разстояние по Левенщайн не са гладки и не могат да се апроксимират. Все пак думите в езика не са случайни стрингове.

Тук е предложена схема вдъхновена от seq2seq модела за превод на думи с рекурентна невронна мрежа [9]. В нашия случай вместо на ниво думи, модела работи на ниво букви.

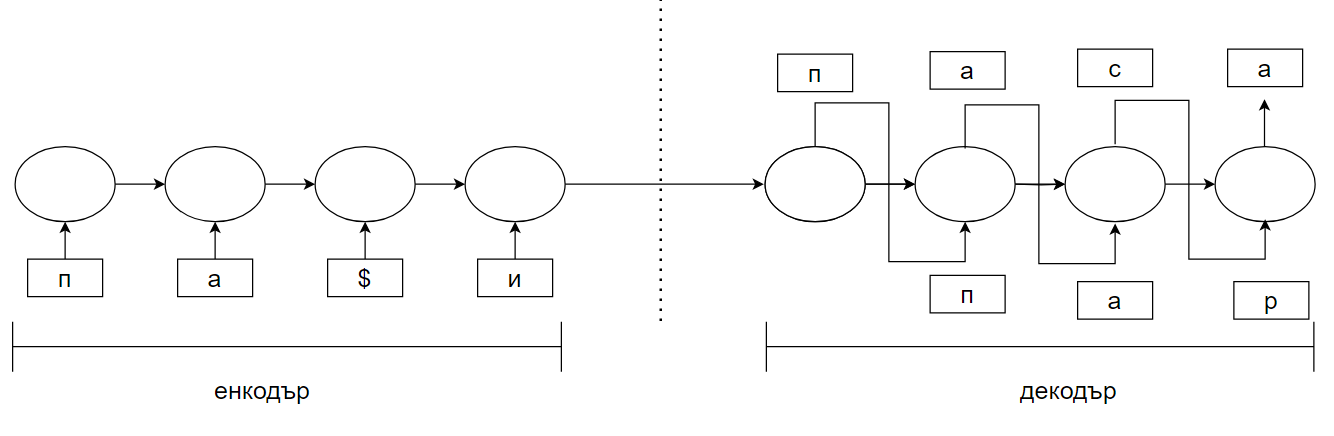
Идеята е да се обучи енкодер-декодер модел на изкуствена задача - изчистване на думите от шум. След обучението желаното влагане бихме получили в скрития вектор произведен от енкодера.

За обучение са използвани думи от автоматично генериран речник от статии на БлицБг, ПикБГ и ДнесБг. Прибавени са също и думите от БГЖаргон. Получени са над 100 000 различни думи. Големия брой на думите дава надежди, че модела няма просто да научи наизуст думите, а ще успее да открие някакви зависимости между буквите, които да отрази в скрития вектор.

Думите се зашумяват автоматично, като им се прилагат една от от следните редакции:

* изтриване на буква
* заместване на буква със случаен символ
* добавяне на случаен символ на случайно място
* мултиплициране на буква няколко пъти
* разместване на 2 букви

Всяка от 100-те хиляди думи е зашумена и от нея са получени по 10. Модела се обучава на тези 1 милион зашумени думи и се тренира чрез метода teacher forcing върху задачата да връща оригиналната дума. (Независимо от предсказаната буква на предходната стъпка, на следващата стъпка като контекст се дава вярната дума). Очаквано не се постигат добри резултати, тъй като премахването на шума не е еднозначна задача, но целта е друга - да се научи енкодера да прави влагане на думите.



Фиг 1. Схема на енкодър декодър модела. В декодъра е показан метода teacher forcing.

Тренира се с 10 епохи. Входния вектор е с размера на азбуката + допълнителни символи. Буквите се кодират чрез one-hot-encoding. Скрития слой е 128-мерен. Алгоритъмът е имплементиран използвайки Pytorch.

След обучение, ембединг на нова дума се получава, като се пусне думата през енкодера и след това скрития вектор се изобразява в по-малко пространство чрез умножение с матрица получена от SVD върху ембедингите на думите от некласифицираните примери. SVD допълнително намалява шума и малко подобрява влагането.

Косинусова близост на векторите, които получават следните низове:

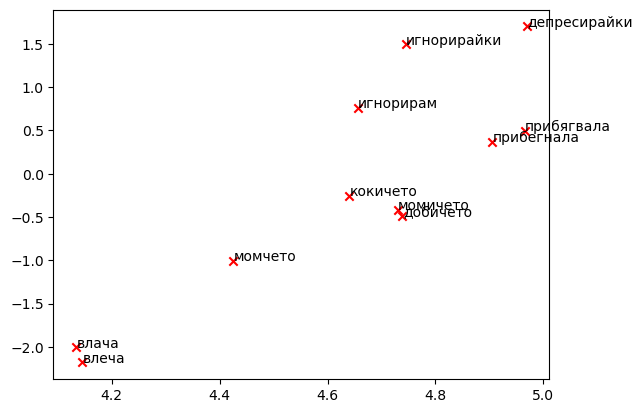
“аааббсс” и “ааббсс” -> 0.9651

“газчотка” и “перваз” -> 0.5395

“абонат” и “таноба” -> 0.7875

“момичето” и “момчето” -> 0.9570

Вижда се, че влагането взима под внимание не само буквите, но и последователността им.

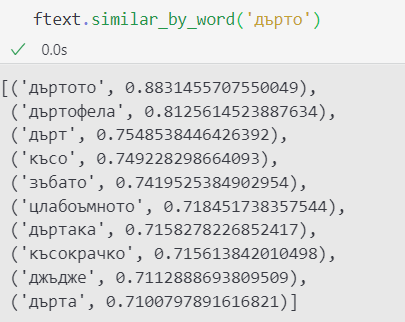


Фиг 2. Векторите на някои думи изобразени в двумерно пространство след SVD

**Векторизация с Fast Text**

Векторизацията с латентен семантичен анализ не включва информация за морфологията на думите. Векторизацията устойчива на шум, пък не включва информация за смисъла на думите. Fast text векторизацията решава тези проблеми, като разбива думите на n-грами, след това прави векторно представяне на всяка n-грама, подобно на word2vec, а влагането на цялата дума е сума от представянията на n-грамите. [10]

Текущата задача за класификация би се възползвала от информация за морфологията на думите, заради многото правописни грешки и непознати думи. Алгоритъма е трениран върху неанотираните данни с 10 епохи. Размера на вектора е 100, минималното срещане на дума за изграждане на речника е 3, размера на прозореца за семантична близост е 5. Използвана е имплементацията от библиотеката Gensim [11].



Фиг 3. 10 най-близки думи до думата ,,дърто’’ според влаганията им от fast text.

Вижда се, че тук има както близки по морфология думи, така и близки по семантика.

**Класификация:**

Задачата за класификация на един коментар се свежда до откриване на мръсна фраза в него - т.е. за класифицирането на коментара не е необходим голям контекст. Това дава идея да се използва конволюционен класификатор.

За оценка на разглежданите класификатори от тук нататък е използвана схемата на k-cross валидацията с k = 10. Представените резултати са осреднени от всеки от 10-те експеримента. Тази схема е избрана, за да се намали влиянието на избора на тестово и тренировъчно множество.

Тестван е конволюционен класификатор с размер на прозореца от 5. Конволюционния класификатор минава през коментара и прилага матрицата на конволюция върху всяка 5-грама. После се прилага максимум от резултатите и този максимум се проектира върху 2-та класа. За имплементацията е използвана библиотеката Pytorch [12].

Конволюционния алгоритъм се научава самостоятелно да разпознава, коя n-грама е мръсна в коментара. Тази информация обаче вече е налична в данните. Една идея да я използваме е да създадем изкуствени коментари от примерите в мръсните коментари. Там гарантирано ще се наблюдава мръсотия и конволюционния класификатор няма да има нужда да учи, коя част от коментара е мръсната.

Резултатите от класификацията са представени в таблицата:

|  |  |  |  |
| --- | --- | --- | --- |
| Влагане | Средна прецизност | Среден  обхват | Средна  F1 оценка |
| под думи | 0.373 | 0.132 | 0.196 |
| шумопоглъщащо | 0.379 | 0.217 | 0.276 |
| поддуми след корекция | 0.315 | 0.346 | 0.330 |
| Fast text | 0.514 | 0.537 | 0.525 |

Таблица 1. Резултати от CNN при съотношение на класовете 1:1

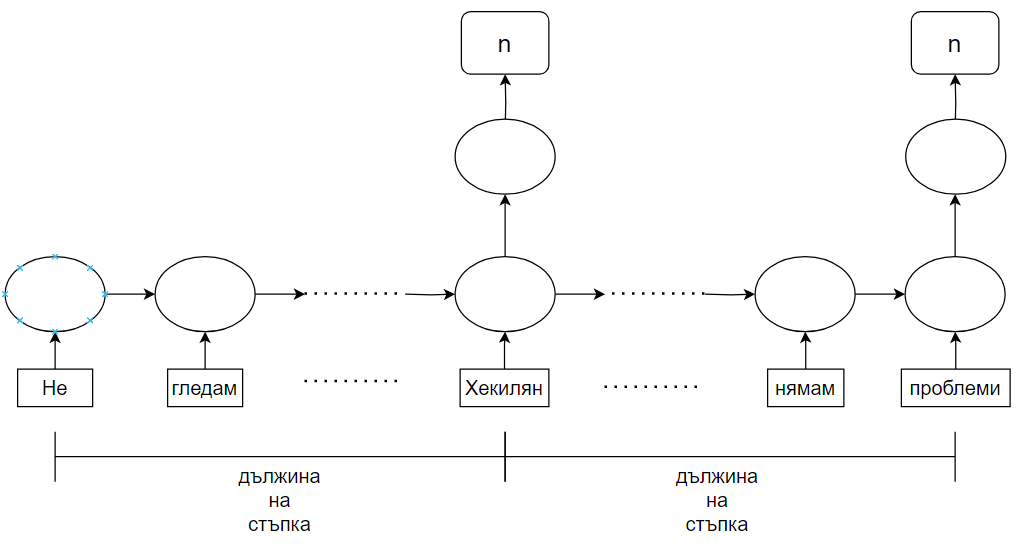
Конволюционния класификатор не се справя със задачата.

**Класифициране с Рекурентна невронна мрежа.**

Ръчната класификация на един коментар протича по следния начин:

* Коментарът се чете отляво надясно и ако се прочете мръсен израз се спира четенето и коментарът се отбелязва за мръсен
* Ако не се намери мръсен израз, коментара се отбелязва за чист.

Директната интерпретация на този процес дава идеята за използването на Рекурентна невронна мрежа. Тя се състои от рекурентен слой, слой за класификация и линеен слой проектиращ в двумерното пространство. За рекурентния слой е използвана LSTM клетка със размерност на скрития вектор и вектора на паметта 100. Класификационния слой има размерност 100, а между него и проектиращия слой e добавена нелинейност ReLu. Алгоритъмът е имплементиран използвайки Pytorch.



Фиг 4. Схема на класификаторът с рекурентна мрежа

Мрежата, в режим за класификация, прочита входа токен по токен и през определен брой стъпки скрития вектор преминава през пълно свързан линеен слой и след това се изобразява в двумерен вектор на класовете. Ако на някоя стъпка се получи класификация на прочетеното като мръсно, то итерирането спира и коментара се обявява за мръсен. В противен случай, ако свърши коментара, то той се класифицира и се връща съответния резултат.

Тренирането на рекурентната мрежа се възползва от примерите, които подсказват, къде има мръсотия в коментара. Всеки коментар е разбит на части от думи. Всяка част съдържа около k на брой думи, където k е стъпката при класификация. Разбиването на части работи така, че нито една дума не е в две съседни части и нито един мръсен израз не е попаднал в две съседни части.

Невронната мрежа изчита всяка разбита последователност и след това изобразява скрития си вектор върху двата класа. Използваната Loss функция е крос ентропия.

Направени са експерименти с размер на стъпката (дължина на класифицираната наведнъж последователност) и със съотношението на класовете при тренирането.

При твърде малък размер на стъпката, например 1, класификаторът дава отклонение в посока класа на мръсните коментари. Това се дължи на факта, че повечето (ако не всички) коментари с дължина една дума са мръсни и класификаторът се научава да отбелязва коментара за мръсен след прочитането на каквато и да е първа дума.

Установено е, че при влагане с корекция на правописа оптимално F1 се получава от стъпка 7 и от съотношение мръсни: чисти около 6-7 към 10.

Резултатите от тестовете на различните влагания са изложени в таблицата:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритъм | Влагане | Средна прецизност | Среден  обхват | Средна F1 оценка |
| RNN | поддуми | 0.525 | 0.596 | 0.558 |
| RNN | шумопоглъщащо | 0.579 | 0.516 | 0.546 |
| RNN | поддуми след корекция | 0.515 | 0.660 | 0.579 |
| RNN | Fast text | 0.680 | 0.742 | 0.710 |

Таблица 3. Резултати при тестове на различни влагания при стъпка 7

и съотношение на мръсни към чисти коментари от 6:10

Вижда се, че Fast Text дава най-добри резултати. Допълнителните изследвания с Fast text влагане върху стъпката и съотношението на мръсни към чисти показват, че най-висок F1 резултат от 0.718 се получава при съотношение 8:10 и стъпка 14.

**Заключение**

За решение на задачата са сравнени различни подходи за векторизация и два алгоритъма за класификация. Наблюдават се по-добри резултати при използване на влагане взимащо в предвид, както семантичната близост, така и морфологията на думите. Очаквано способността да се гледа по-широк контекст, допринася за класификацията.

Не са постигнати задоволителни резултати, като основна причина за това е ограниченото количество данни и трудността на задачата. Качеството на текста е твърде ниско, а потребителите са много оригинални в грубостите си.

Въпреки това, курсовата работа дава някои идеи за класифициране на текстове с ниско качество.

**Бъдеща работа**

Идея, която не намери място в експериментите на текущия проект, е да се ползват големи модели от типа на Bert. Възможно е да се постигнат по-добри резултати, ако се използва претрениран Bert за български език, който е дотрениран върху неанотираните коментари на същата задача като при претренирането и след това се дотренира за класификация на анотираните. Възможно е и разширение на речника на токенизатора му с такъв получен от неанотираните коментари.

**Използвана литература и източници**

**Първа част**

[1] Levenshtein, Vladimir I. (February 1966), "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady*

[2] Damerau, Fred J. (March 1964), "A technique for computer detection and correction of spelling errors", *Communications of the ACM*

[3] К-gram indexes for spelling correction https://nlp.stanford.edu/IR-book/html/htmledition/k-gram-indexes-for-spelling-correction-1.html

[4] Jaccard, Paul (February 1912). "The Distribution of the Flora in the Alpine Zone.1". *New Phytologist*.

[5] SymSpell <https://github.com/wolfgarbe/SymSpell>

**Втора част**

[6] Wordpiece tokenization <https://huggingface.co/learn/nlp-course/en/chapter6/6>

[7] Latent Semantic Analysis <https://scikit-learn.org/stable/modules/decomposition.html#truncated-singular-value-decomposition-and-latent-semantic-analysis>

[8] TruncatedSVD <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD>

[9] Seq2Seq translation tutorial <https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html>

[10] Enriching Word Vectors with Subword Information Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov

[11] Gensim Fast Text <https://radimrehurek.com/gensim/models/fasttext.html>

[12] Pytorch <https://pytorch.org/> и документация на Pytorch <https://pytorch.org/docs/2.2/>

**Връзка към хранилище**

<https://github.com/NikolayDPaev/profanity-detector>