КУРСОВ ПРОЕКТ

Свързване на обекти с Магелан

|  |  |
| --- | --- |
| **Факултет по Математика и Информатика** **Студент:** Борислав Стоянов Марков **Факултетен номер:** **0MI3400048** **Учебен план:** **Изкуствен Интелект (редовно, магистър)** Курс: **Курс 1**; Група: **Група 1** **Активен период**: 2021/2022 летен, магистри **Дисциплина**: Семантичен Уеб | **Факултет по Математика и Информатика** **Студент:** Кирил Димов Георгиев **Факултетен номер:** **1MI3400098** **Учебен план:** **Изкуствен Интелект (редовно, магистър)** Курс: **Курс 1**; Група: **Група 1** **Активен период**: 2021/2022 летен, магистри **Дисциплина**: Семантичен Уеб |

# 1. Съдържание

[1. Съдържание 2](#_Toc103890517)

[2. Увод 2](#_Toc103890518)

[3. Реализация 3](#_Toc103890519)

[3.1 Алгоритъм 3](#_Toc103890520)

[3.1.1 Процес на свързване на обекти 4](#_Toc103890521)

[3.2 Библиотека Магелан 6](#_Toc103890522)

[3.3 Корпус с данни 7](#_Toc103890523)

[3.4 Редукция на входните множества (blocking) 8](#_Toc103890524)

[3.5 Свързване на обектите 9](#_Toc103890525)

[3.5.1 Подобряване на свързването с нови полета 10](#_Toc103890526)

[3.6 Използване на приложението 12](#_Toc103890527)

[3.7 Оценка на резултатите 13](#_Toc103890528)

[4. Недостатъци и подобрения 13](#_Toc103890529)

[5. Източници и използвана литература 14](#_Toc103890530)

[Приложения 15](#_Toc103890531)

[1. Сорс код (Source code) 15](#_Toc103890532)

# 2. Увод

Свързването на единици от различни множества е често срещан проблем при колективните онлайн магазини. Например различни доставчици на данни подават към онлайн платформите данни въведени от човек, но в слабо структурирана форма. Да кажем Amazon.com продава една и съща стока но от различни търговци. Много е важно да има алгоритъм по който да се намира вече въведената стока дали я има в онлайн магазина. По същият начин има държавни институции, които имат данните за населението но от различни източници и трябва да се засече кои индивиди имат повече от един адрес или са декларирали невярна информация, както и да се намерят различните такива за да се преброи населението коректно.

# 3. Реализация

Има много сценарии по които може да се реши дали няколко различни обекта отговарят на един и същ обект от реалния свят. В случая сме избрали да слеем две таблици в една. Данните ще получим като CSV файлове. Ще използваме данни предоставени от Анхайс Груп(AnHai's Group), събрани от студенти [3], служещи за демонстрация на проекта Магелан. В случая става въпрос за компютърна техника, събрана от американските сайтове Amazon и Best Buy. Данните са във форма на CSV и имат следният вид:

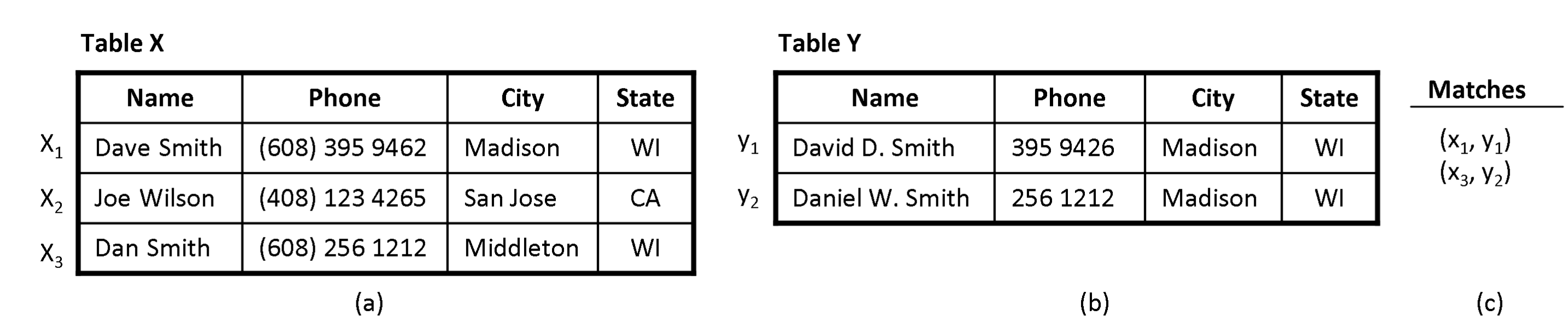
|  |
| --- |
|  |

Таблица 3.1

За реализация сме избрали проектът на Анхайс Груп наречен Магелан(Magellan) [1],[2], написан на Python. Изпълнимият код ще предоставим като Юпитер Ноутбук(Jupyter Notebook) файлове, за които има инструкции в Приложението.

## 3.1 Алгоритъм

Има различни сценарии да кажем кои обекти представляват един и същи обект от реалния свят. В случая по-формално можем да кажем, че имаме две таблици A и B. Искаме да намерим всички наредени двойки (a,b) от множествата, които отговарят на един и същ обект от реалността. На фигура 3.1.1 сме представили процеса нагледно.



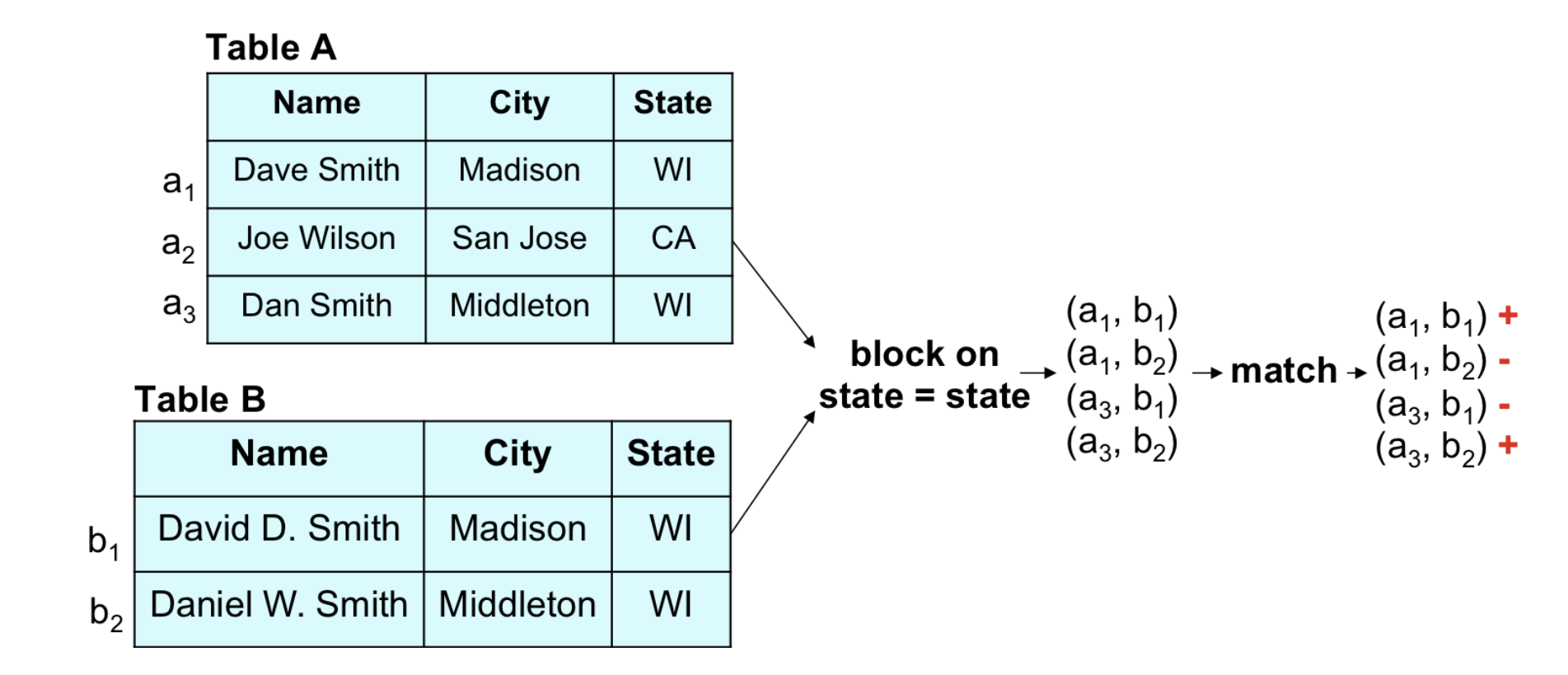
Фигура 3.1.1 Свързване на обекти, източник: [1]

### 3.1.1 Процес на свързване на обекти

В практиката този процес е на два етапа:

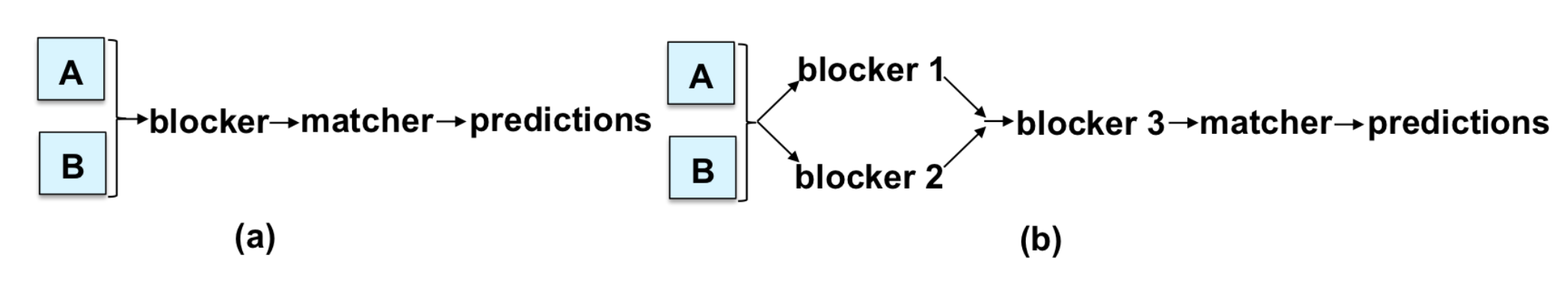
* Изисквания: според нуждите на бизнеса се уточнява специфични правила по които да се свържат обектите и да се постигне висока точност (precision и recall, вж. [6]).
* Разработка: с даден корпус от данни се опитва да се постигне висока точност по зададени критерии според изискванията на бизнеса.

Тъй като имайки две множества A и B, за да вземем всички възможни двойки, то това би било тяхното декартово произведение A x B ={ (a, b) | a∈A, b∈B } би било твърде голямо като обем. За целта се прави редукция или т.нар. **blocking.** Следтова, вече на редуцираното множество се прави реалният процес по свързване (**matching**). Този процес е означен на фигура 3.1.1.1. Плюс и минус знаците най-отдясно означават кои двойки са одобрени от процеса по свързване.

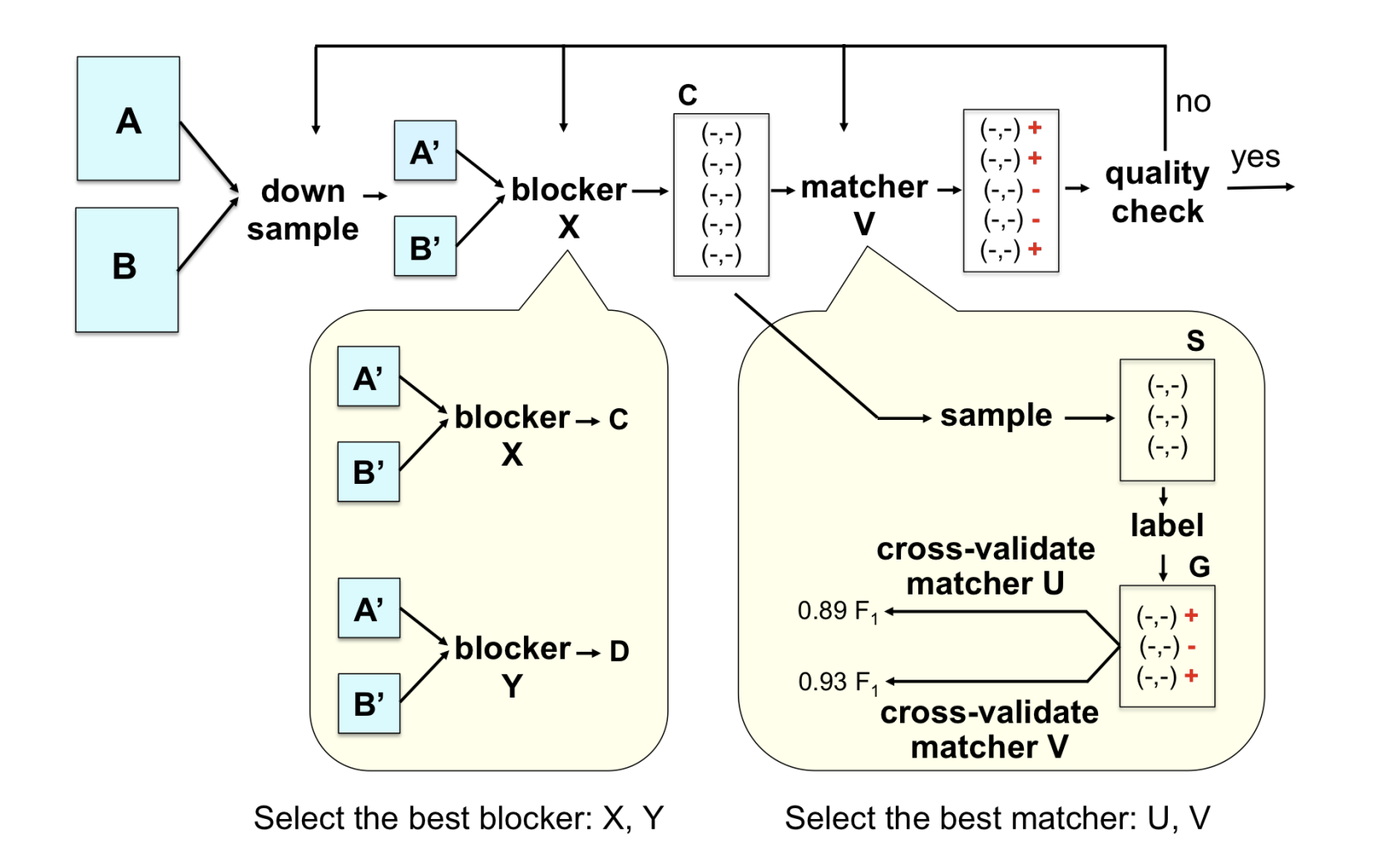


Фигура 3.1.1.1. Редукция и свързване на обекти, източник: [1]

Процесът по редукция, може допълнително да се раздели ня няколко вида. Примерни варианти са дадени на фигура 3.1.1.2. На фигурата, подточка (а) е показана плоска структура и само един редуктор. На подточка (b) са изобразени два редуктора **blocker 1** и **blocker 2**,работещи независимо един от друг и после обединеното множество се редуцира допълнително от **blocker 3** и преминава през свързване с **matcher** и правене на предположение.

 Фигура 3.1.1.2. Редукция на множества, източник: [1]

След като са обработени първоначално двете множества, новите атрибути са добавени и са използвани различни блокинг техники за създаване на кандидатите, трябва обектите от двете множества да бъдат свързани. Свързването на обектите се осъществява посредством алгоритъм от областта на машинното самообучение, който трябва да предскаже, с някаква вероятност, дали дадени два обекта са едни и същи. За да получим максимален брой точни свързвания, трябва да бъде избран подходящ алгоритъм, който да бъде максимално добре обучен, върху тренировъчните данни и да бъде изпробван върху тестовите данните. Ще бъде използван подходът ,,К-Fold Cross Validation” показан на фигура 3.1.1.3.



Фигура 3.1.1.3

Избирането на най-добрия алгоритъм е според метриката „точност“ (precision) [6] за измерване на алгоритмите. Алгоритмите, измежду които ще бъде избирано, са „дърво на решенията“ (Decision Tree), „support vector machine“, „случайна гора“ (random forest), „логистична регресия“ и „линейна регресия“ (logistic and linear regression). След това изчислителното множество (evaluation set) бива превръщан във фийчър вектори за тренировъчните и тестовите данни. Сега биват използвани фийчър векторите от тренировъчния сет, за да бъде трениран алгоритъма, и го изпробваме върху тестовото множество.

## 3.2 Библиотека Магелан

Библиотеката Магелан е разработена от групата Анхайс [1]. Тя е написана на програмния език Python и използва компоненти написани на език C от по-ниско ниво. Само така може да осигури исканата бързина. Библиотеката дава набор от редуктори(блокери) [5], които можем да дадем в таблица таблица 3.2.1.

|  |  |
| --- | --- |
| AttrEquivalenceBlocker | Редуцира или оставя тези двойки (a,b) , които имат еднаква стойност на атрибут |
| OverlapBlocker | Редуцира на базата на една или няколко съвпадащи думи или q-грами. [5] |
| RuleBasedBlocker | Работи на базата на потребителска селекция от правила, които се изпълняват за всяка една наредена двойка. |
| BlackBoxBlocker | Работи на базата на функция, която връща истина или неистина, функцията се дава от потребителя и се изпълнява за всяка наредена двойка. |

Таблица 3.2.1

Библиотеката Магелан предоставя и набор от алгоритми за машинно самообучение, които могат да бъдат използвани за свързване на обектите. Te са следните:

1. дърво на решенията (DecisionTree)
2. случайна гора (Random Forest)
3. машина на поддържащи вектори (Support Vector Machine)
4. линейна регресия (LinReg)
5. логистична регресия (LogReg)

Също така този фреймуорк предоставя функцията ,,select\_matcher”, която чрез ,,К-Fold Cross Validation” метода избира най-добрия алгоритъм, който да бъде използван. За да може да бъде трениран алгоритъма, данните от суров вид трябва да бъдат преобразувани във фийчър вектори. Това става посредством фукцията „extract\_feature\_vecs“.

## 3.3 Корпус с данни

Групата Анхайс има специално подбрани множества от данни над които може да се тества библиотеката им. Връзка е дадена в [3]. Изглед е даден на таблица 3.3.1.

|  |
| --- |
|  |

Таблица 3.3.1

Избрали сме да използваме номер 8 за електроника поради няколко причини. Едната е, че имаме опит с лаптопи и дребна електроника, втората причина е, че текстовото описание не е голямо и няма да изисква обработка на естествен език (NLP) и като размер двете множества са сравнително неголеми и немалки. От посочените данни използваме само „A“ и „B“ от секцията „Input Tables“. Не изпозлваме анотираните данни от “Labeled Data” – “L”, тъй като сме избрали свой подход за редукция(blocking) и тези анотирани данни вероятно няма да са в нашето редуцирано множество и от друга страна открихме твърде свободно анотирани данни, което не считаме за правилно.

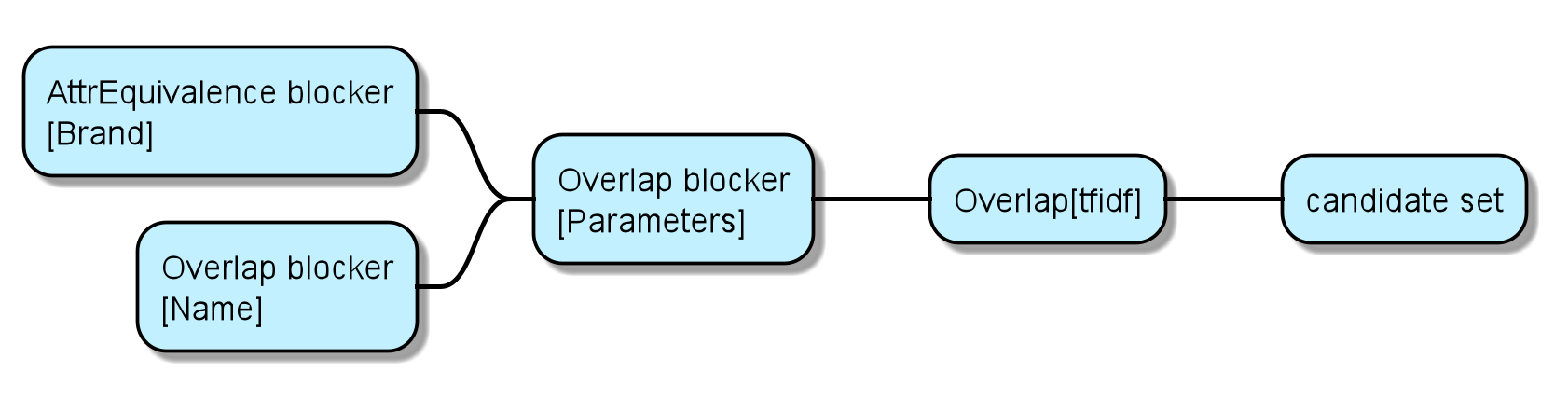
|  |  |
| --- | --- |
|  |  |

Таблица 3.3.2

Разархивирани данните и колоните са показани схематично в таблица 3.3.2. Виждаме, че някои от колоните нямат еднозначно съответствие.

## 3.4 Редукция на входните множества (blocking)

За конкретния случай в зависимост от колоните на двете таблици сме избрали редуктори, които са формирани на базата на това колко са качествени избраните данните и какви колони имаме в табличните данни. В началото забелязваме, че марката съвпада на повечето лаптопи и друга техника, например “Asus”, “Apple” и други имат еднакво изписване в „A“ и „B“. Mожем да кажем, че искаме тези атрибути да са еднакви. Също така искаме и “Name” да има някакво препокриване от поне две думи. След това обединяваме множествата получени с тези два редуктора и допълнително редуцираме на базата на други параметри, които сме добавили ние. Схематично това е показано на фигура 3.4.1



Фигура 3.4.1. Избрани редуктори

Параметрите, които сме добавили се наричат „Parameters” и “tfidf”. Чрез графично представяне на данните, се улеснява изпълнението на следващите стъпки и се показва какви термини се срещат. Разпределението е сравнително равномерно и няма доминиращи термове.

A picture containing text, measuring stick

Description automatically generated

Фигура 3.4.2 Честота на срещанията на А

A picture containing histogram

Description automatically generated

Фигура 3.4.3 Честота на срещанията на B

Считаме, че при описанието на компютрите цифрите имат значение, дали е 4GB, дали процесора е i5 или модела е UX305A, това дава една характеристика, която може да ни помогне да редуцираме допълнително първоначалната селекция. Втория параметър сме именовали “tfidf” [4] защото това е една статистика за честотата и значението на всеки терм към неговия документ (запис в таблицата). Това ни помага да извлечем статитически значимите термове от описанието за лаптопи и друга дребна техника. Например можем да видим как тази статистика веднага намира подходящите отличителни белези на съответния лаптоп от следния пример, даден на Таблица 3.4.1.

|  |
| --- |
|  |

Таблица 3.4.2

Вземат се първите 5 значими терма и това обикновено е модела на лаптопа(x205ta), дисплея в инчове (11 на втория ред) или някакво име на серия или търговско име(zseries -втори ред ,chromebook – 4-ти ред).

Имайки вече редуцираното множесто от продукта на множествата A и B можем да извадим на случаен принцип една представителна извадка от 500 примера, които да **анотираме** ръчно и да кажем кои елементи от лявата страна съответстват на дясната страна. Така можем да натренираме някакъв алгоритъм или няколко алгоритъма, които ще се използват вече за самото съответствие. Ръчно сме анотирали данните във файла „sample\_blocked\_500\_labeled.csv“ и колоната се казва “label”, като приема стойност „0“ за несъвпадение и „1“ за съвпадение. Именно така можем да продължим със свързването на обектите.

## 3.5 Свързване на обектите

Трябва да заредим двете нови таблици с новите колони, както и **анотираните** данни. Следващата стъпка от алгоритъма е превръщано на изчислителното множество във фийчър вектори за тренировъчните и тестовите данни. Анотираните данни са разделени на обучаващо и тестово множество в пропорция 75% към 25%.

Ще бъде използван подходът ,,К-Fold Cross Validation” върху обучаващото множество, за да изберем най-добрият алгоритъм, като ще пробваме различни стойности на k и метриката за измерване на алгоритмите – точност (precision). Алгоритмите, измежду които ще бъде избирано, са: Дърво на решенията (Decision Tree), машина на поддържащите вектори (support vector machine), случайна гора (random forest), логистична и линейна регресия (logistic and linear regression). След изпълнението на ,,К-Fold Cross Validation” получените резултати показват, че алгоритъмът „дърво на решенията“ се справя най-добре и това е показано на таблица 3.5.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | Decision Tree/Precision | Random Forest/Precision | LinReg/Precision | LogReg/Precision |
| 11 | **0.48** | 0.27 | 0.00 | 0.09 |
| 10 | **0.55** | 0.30 | 0.10 | 0.20 |
| 9 | **0.57** | 0.36 | 0.11 | 0.11 |
| 8 | **0.60** | 0.38 | 0.13 | 0.19 |
| 7 | **0.50** | 0.57 | 0.14 | 0.14 |
| 6 | **0.48** | 0.33 | 0.17 | 0.08 |
| 5 | **0.52** | 0.33 | 0.2 | 0.23 |

Таблица 3.5.1

Сега използваме фийчър векторите от тренировъчния сет, за да тренираме алгоритъма, и го изпробваме върху тестовото множество. Получаваме следните резултати:

|  |
| --- |
| Precision : 54.55% (6/11)  Recall : 30.0% (6/20)  F1 : 38.71%  False positives : 5 (out of 11 positive predictions)  False negatives : 14 (out of 139 negative predictions) |

Резултатите не са добри и ще търсим подобрение.

### 3.5.1 Подобряване на свързването с нови полета

За да бъдат подобрени резултатите, ще бъдат добавени нови атрибути при тренирането и оценяването на кандидате, получени от предишните стъпки. Това е атрибутът цена (“Price“) и атрибутът, които описва инчовете на електронните устройства (“Screen”). За тази цел са създадени две функции: ,,clean\_price“ и ,,guess\_screen\_size“. Първата функция взема текстовият низ, който репрезентира цената на продукта, и я превръща в число. Втората функция съединява текстовите полета: името, описанието и характеристиките на продукта и търси определени числа (пример 17.3) и в случай, че намери някое от изброените числа, то продуктът получава стойност за инчовете. Продукти, които нямат намерени инчове, считаме, че нямат стойност. Разпределението на цената е дадено в таблица 3.5.1.1. Виждаме, че средната цена на лаптоп е около $243 за „А“ и $352 за „B”. Хистограмата показва, че в първото множество имаме в класа над $100 преобладават лаптопи на цена от $500, докато във второто множество в класа на $100 преобладават лаптопи на цена $100-$200.

|  |  |
| --- | --- |
| Разпределение на цената за „А“ от $0-$100 и от $100 нагоре. Ред 2 дава статистически характеристики на цената. | Разпределение на цената за „B“ от $0-$100 и от $100 нагоре. Ред 2 дава статистически характеристики на цената. |
|  |  |
|  |  |

Таблица 3.5.1.1

Можем да разгледаме и как са разпределени лаптопите по инчове на екраните, дадени в таблица 3.5.1.2.

|  |  |
| --- | --- |
| Хистограма на размер дисплеи за множество „А“ | Хистограма на размер дисплеи за множество „B“ |
|  |  |

Таблица 3.5.1.2

Виждаме, че групата „none“, т.е. без размер е доста голяма и може да доведе до грешни резултати. Вероятно това са захранвания, процесори или други компоненти без дисплей.

След пускането на дърво на решенията върху цялото тренировъчно множество за трениране на модела той получава по-добри резултати върху тестовото множество, а те имено са:

|  |
| --- |
| Precision : 66.67% (8/12)  Recall : 80.0% (8/10)  F1 : 72.73%  False positives : 4 (out of 12 positive predictions)  False negatives : 2 (out of 113 negative predictions) |

Виждаме, че добавените полета са подобрили значително всички параметри.

## 3.6 Използване на приложението

Приложението може да бъде използвано, след като се изпълнят следните стъпки:

1. За клониране на репозиторито в „git bash” терминала изпълняваме командата git clone <https://github.com/borkox/uni-sofia-entity-linking-magellan.git> в избраната директория. По желание можем да инсталиране на *anaconda* приложението. Документация за инсталация може да да бъде намерен в [7].

2. След като бъде инсталирана anaconda, чрез anaconda трябва да бъде отворен терминал:

|  |
| --- |
| Graphical user interface, application  Description automatically generated |

Таблица 3.6.1

3. След това в терминала изпълняваме: „pip install -U numpy scipy py\_entitymatching“. Инсталацията е описана също и в Приложение.

Структурата на приложението и работа с Jupyter са изобразени на следващата фигура.

|  |  |
| --- | --- |
|  |  |

Таблица 3.6.2

## 3.7 Оценка на резултатите

|  |
| --- |
| # Evaluate the predictions  eval\_result = em.eval\_matches(predictions, 'label', 'predicted')  em.print\_eval\_summary(eval\_result) |
| Precision : 66.67% (8/12)  Recall : 80.0% (8/10)  F1 : 72.73%  False positives : 4 (out of 12 positive predictions)  False negatives : 2 (out of 113 negative predictions) |

За да оценим реално резултатите, трябва моделът да бъде тестван върху целия корпус от кандидати, чиято бройка е 282581. Зареждаме всички кандидати от ,,csv” файла и чрез функцията ,,predict” бива предсказано, дали дадена двойка кандидати съответстват на един и същи продукт. Резултатът показва, че 22615 е общият брой на съвпадения, но причината да са толкова много, е че огромна част от двойките се повтарят или отляво или отдясно. Затова трябва да филтрираме по идентификационния номер на една от двете първоначални таблици. След филтрация крайният брой е 558.

# 4. Недостатъци и подобрения

В момента сме разработили само девелопмънт частта от процеса на свързване. Не е разбработена частта, която може да се деплойне и не сме посочили как данните да идват в поточен вид. Друг недостатък е, че избрания алгоритъм зависи от данните, а това е нежелано.

Можем да се подобри свързването на обектите от тип захранване и да се извлече тяхната мощност с предварителна обработка на данните. По същият начин можем да извлечем специфични колони за калъфите за лаптопи. Като по-общо решение можем да сложим една колона, която да определя категорията на продукта, лаптоп, захранване, калъф и др.

# 5. Източници и използвана литература

[1] How-To Guide to Entity Matching, AnHai's Group, 2017   
<https://pradap-www.cs.wisc.edu/magellan/how-to-guide/how_to_guide_magellan.pdf>

[2] User Manual for py\_entitymatching, AnHai's Group, 2017   
<http://anhaidgroup.github.io/py_entitymatching/v0.3.x/index.html>

[3] The 784 Data Sets for EM, students in the CS 784 data science class at UW-Madison, 2015  
<https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository>

[4] tf–idf - Wikipedia  
<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

[5] API reference for Blockers, An Haim Group

<http://anhaidgroup.github.io/py_entitymatching/v0.1.x/user_manual/api/blocking.html>

[6] Accuracy, Precision, Recall or F1?, Koo Ping Shung, Mar 15, 2018

<http://anhaidgroup.github.io/py_entitymatching/v0.1.x/user_manual/api/blocking.html>

[7] Anaconda installation guide

<https://docs.anaconda.com/anaconda/install/>

# Приложения

## 1. Сорс код (Source code)

Кодът е публичен и качен в платформата Гитхъб.

<https://github.com/borkox/uni-sofia-entity-linking-magellan/>