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  **Дисциплина**: Откриване на информация в данни |

# 1. Съдържание

[1. Съдържание 2](#_Toc103542121)

[2. Увод 3](#_Toc103542122)

[3. Реализация 3](#_Toc103542123)

[3.1 Алгоритъм 3](#_Toc103542124)

[3.2 Библиотека 3](#_Toc103542125)

[3.3 Корпус с данни 3](#_Toc103542126)

[3.4 Редукция на входните множества 3](#_Toc103542127)

[3.4.1 Допълнително редуциране с нови полета 3](#_Toc103542128)

[3.5 Свързване на обектите 3](#_Toc103542129)

[3.5.1 Подобряване на сравнението с нови полета 3](#_Toc103542130)

[3.6 Използване на приложението 3](#_Toc103542131)

[3.6 Оценка на резултатите 3](#_Toc103542132)

[4. Недостатъци и подобрения 3](#_Toc103542133)

[5. Източници и използвана литература 3](#_Toc103542134)

[Приложения 3](#_Toc103542135)

[1. Сорс код (Source code) 3](#_Toc103542136)

# 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 Анхайм Груп, 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:

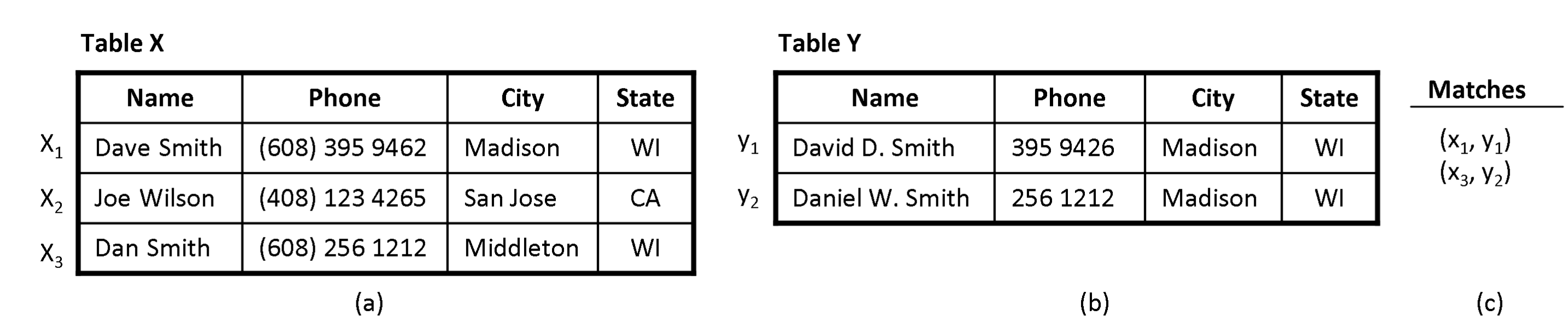
|  |
| --- |
|  |

Таблица 3.1

За реализация сме избрали проектът на Анхайм Груп наречен Магелан [1],[2], написан на Python. Изпълнимият код ще предоставим като 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 1 сме представили процеса нагледно.



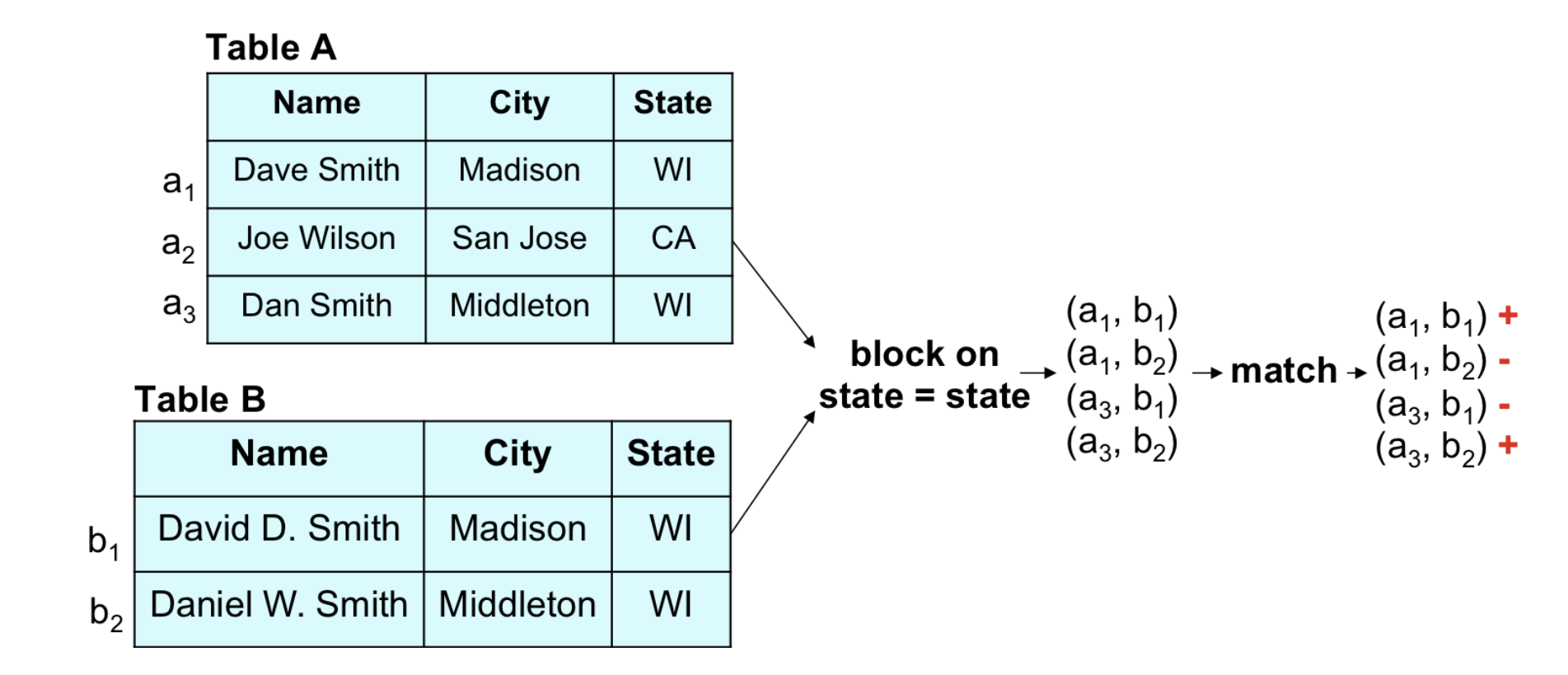
Фигура 3.1.1 Свързване на обекти, източник: [1]

### 3.1.1 Процес на свързване на обекти

В практиката този процес е на два етапа:

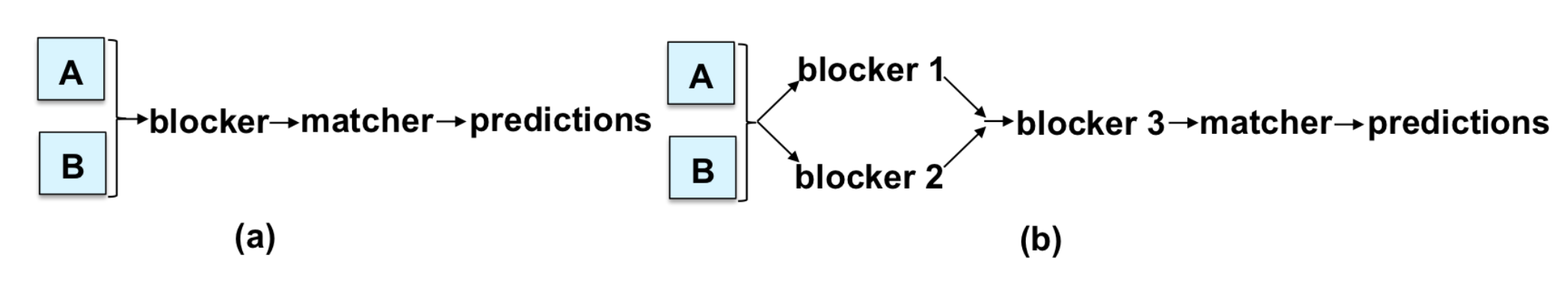
* Изисквания: според нуждите на бизнеса се уточнява специфични правила по които да се свържат обектите и да се постигне висока точност precision и recall.
* Разработка: с даден корпус от данни се опитва да се постигне висока точност по зададени критерии според изискванията на бизнеса.

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 3.1.2. Плюс и минус знаците най-отдясно означават кои двойки са одобрени от процеса по свързване.

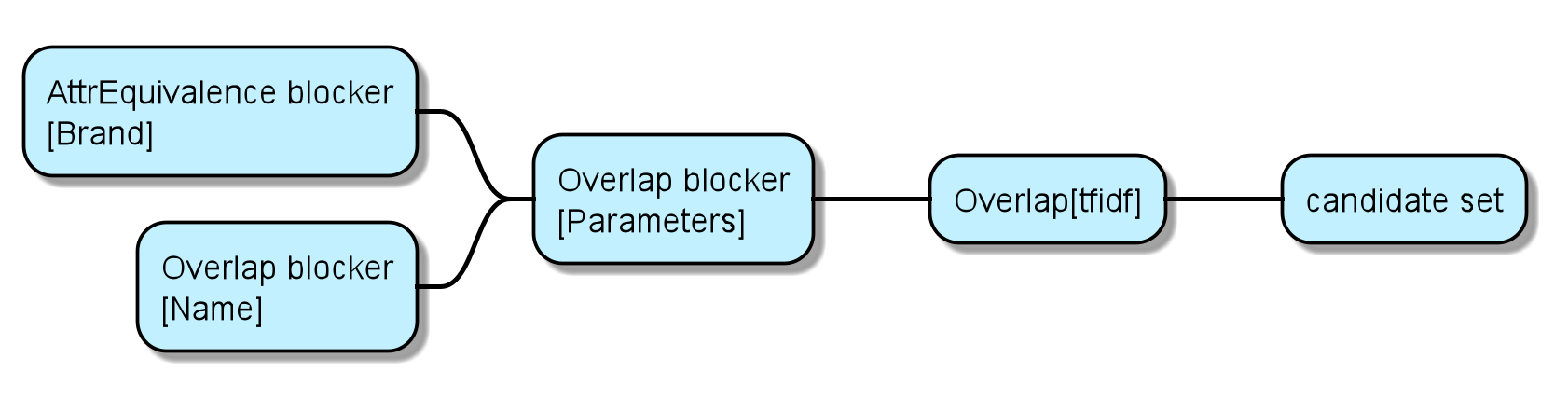


Фигура 3.1.2. Редукция и свързване на обекти, източник: [1]

Процесът по редукция, може допълнително да се раздели ня няколко вида. Примерни варианти са дадени на фигура 3.1.3. 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 3.1.3. Редукция на множества, източник: [1]

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 съдържание. Значи можем да кажем, че искаме да са еднакви тези атрибути. Отделно искаме и “Name” да има някакво припокриване 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 3.1.4



Фигура 3.1.4. Избрани редуктори

Параметрите, които сме добавили се наричат „Parameters” и “tfidf”. 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 кръстили “tfidf” [4] защото това една статистика за честотата и значението на всеки термин 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 3.1.1.1.

|  |
| --- |
|  |

Таблица 3.1.1.1

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.

**KIRIL TODO**

## 3.2 Библиотека Магелан

Библиотеката Маелан е разработена от групата Анхайм [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

**KIRIL TODO**

## 3.3 Корпус с данни

Групата Анхайм 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 невролингвистично програмиране (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 н вероятно няма да са в нашето редуцирано множество и от друга страна открихме твърде свободно анотирани данни, което не считаме за правилно.

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

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

Таблица 3.3.2

## 3.4 Редукция на входните множества

### 3.4.1 Допълнително редуциране с нови полета

Допълнителните полета за редукция както беше описано в алгоритмичната част се получават по следния начин както е дадено в таблица 3.4.1

|  |
| --- |
| **def** filter\_tokens\_with\_digits(s):  s **=** str(s)  s **=** re**.**sub('&#34;|[\*\\\()\-\/]', ' ',s)  s **=** re**.**sub(',', ' ',s)  toks **=** s**.**split()  toks **=** list(filter(**lambda** x: bool(re**.**search(r'\d', x)), toks))  **return** ' '**.**join(toks)  A['Parameters'] **=** A['Name'] **+** " " **+** A['Features']  A['Parameters'] **=** A['Parameters']**.**apply(filter\_tokens\_with\_digits)  ...  **from** sklearn.feature\_extraction.text **import** TfidfVectorizer  **import** numpy **as** np  tf **=** TfidfVectorizer(input**=**'content', analyzer**=**'word', ngram\_range**=**(1,1),  min\_df **=** 0, stop\_words **=** 'english', sublinear\_tf**=True**)  *# TFIDF for set A*  tfidf\_matrix **=** tf**.**transform((A['Brand'] **+** ' ' **+** A['Name'])**.**fillna(""))  feature\_array **=** np**.**array(tf**.**get\_feature\_names())  **def** extract\_top\_tfidf(column):  response **=** tf**.**transform([column])  tfidf\_sorting **=** np**.**argsort(response**.**toarray())**.**flatten()[::**-**1]  n **=** 5  top\_n **=** feature\_array[tfidf\_sorting][:n]  *# stringify*  **return** " "**.**join(top\_n)  A['tfidf'] **=** ((A['Brand'] **+** ' ' **+** A['Name'])**.**fillna(""))**.**apply(extract\_top\_tfidf)  B['tfidf'] **=** ((B['Brand'] **+** ' ' **+**B['Name'])**.**fillna(""))**.**apply(extract\_top\_tfidf) |

Таблица 3.4.1

## 3.5 Свързване на обектите

### 3.5.1 Подобряване на свързването с нови полета

## 3.6 Използване на приложението

## 3.6 Оценка на резултатите

# 4. Недостатъци и подобрения

# 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>

# Приложения

## 1. Сорс код (Source code)

Кодът е публичен и качен в платформата Гитхъб.

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