Факултет инжењерских наука

Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Предмет: Рачунарски алати

Број индекса: 563/2015

Никола П. Марковић

Рачунарски алат за једноставну обраду текстуалних докумената заснован на командној линији

дипломски рад

Комисија за преглед и одбрану:

1. **доц. др Владимир Миловановић - ментор**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

У оквиру овог завршног рада кандидат треба да:

Имплементира рачунарски алат који се покреће из командне линије и као улаз прима текстуалну датотеку екстензије *.txt* и *.docx*. Неопходно је направити помоћну наредбу *-p/--помоћ* која ће кориснику дати јасан преглед свих могућности које наведени алат нуди. Након извршених жељених измена, излазна датотека се може снимити под екстензијом коју има и улазна (измењена) датотека или пак у *.pdf* формату.

Крагујевац, 16.05.2019. год.

ментор:

доц. др Владимир Миловановић

_____

Резиме рада:

Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата - *Python, Ghostscript, Paps, LibreOfficeWriter, OfficeToPdf*. Пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су **регуларни изрази**, **Левенштајново растојање**. Након што се читалац упозна са технологијама и појмовима које је потребно знати, следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљно објашњење везано за њено покретање.

Кључне речи:

Рачунарски алат, текстуална датотека, *Python*

Abstract:

This paper begins with brief introduction about technologies that were used for creating final version of sotfware tool - *Python, Ghostscript, Paps, LibreOfficeWriter, OfficeToPdf*. Firstly reader must be inform about few terms like **regular expressions**, **Levenshtein distance**. After that, reader can fully understand the next part which describe all parts of software tool logic. In last part of this paper is brief explanation about usage of this software tool.

Key words:

Software tool, text file, *Python*

Садржај

Увод

Како живимо у времену дигитализације где је све више грана индустрије повезано са технологијом, тако расту и потребе за дигиталном писменошћу запослених. Писање извештаја, докуемнтације или било какве врсте текста, постала је свакодневница. Како саме текстуалне датотеке могу да буду обимне и заморне за проверу евентуалних грешака, често се дешава да исте буду откривене касно.

При самом вршењу провере текста, треба водити рачуна и о следећим стварима:

1. Један од могућих пропуста јесте мало почетно слово у реченици односно после тачке.
2. Приликом брзог куцања може се изоставити размак између запете (,) и неке речи односно тачке (.) и неке речи.
3. Такође, корисник случајно може да унесе више размака између две речи или између речи и знакова као што су запета (,) и тачка (.) иако је желео само један размак између наведених секвенци.
4. Неретка је појава да је потребно извршити конверзију из латиничног писма у ћирилично и обрнуто.
5. Веома честа појава јесте да се у самом куцању јави погрешно написана реч где је корисник случајном грешком додао слово или знак који су сувишни (нпр. наукас уместо наука).
6. Некада је потребно заменити појављивање специфичне секвенце у тексту неком новом секвенцом знакова.

Тематика овог завршног рада се базира управо на корекцији одређених грешака које су тешко уочљиве на први поглед. Осим опције за корекцију текста, софтверски алат нуди још три функционалности:

- Информативну:
    1. Корисник специфицира број најчешће коришћених речи које се налази у тексту и након тога следи приказ истих.
- Конверзија текстуалних датотека у *.pdf* формат:
    1. Датотека екстензије *.txt* у *.pdf*
    2. Датотека екстензије *.docx* у *.pdf*

Технологије

Програмски језик *Python*

Слика 1: Лого програмског језика *Python*

Представља интерпретирани програмски језик, високог нивоа. Прва верзија овог све популарнијег програмског језика настала је 1991. године а његов оснивач јесте Гвидо Ван Росум. *Python* није типизиран програмски језик за разлику од других програмских језика као што су *Java* или *C++* . Он аутоматски након додељивања вредности променљивој процењује који је тип променљива (ниска, број са покретним зарезом, реалан број итд.). Подржава више парадигми из света програмирања као што су објектно, процедурално или функционално програмирање. У шали се каже да је *Python* константно „Укључен у напајање“ јер садржи веома велики број библиотека. [1]

Основна сврха наведеног програмског језика јесу послови аутоматизације и скриптовања. Све више се користи у доменима науке о подацима, машинског учења, развоју веб апликација. У овом програмском језику могуће је креирати и *GUI* (графкички кориснички интерфејс) и *CLI* (интерфејс командне линије) апликације. Сам софтверски алат који је тема овог рада представља *CLI* тип апликације.

Библиотеке програмског језика *Python* које су коришћене у самом креирању софтверског алата су:

- ***Argparse***
- ***Re***
- ***Docx***
- ***Platform***
- ***Numpy***
- ***PySpellChecker***

Библиотека *Argparse*

Наведена библиотека омогућава да се у оквиру програмског језика *Python* креирају скрипте које су базиране на командној линији. Прва верзија ове библиотеке објављена је 20. фебруара, 2011. године. [2]

Омогућава следеће ствари:

- Коришћење позиционих аргумената (аргументи су утврђени њиховим редоследом навођења)
- Омогућава измену знакова за префикс наведеног аргумента (нпр. уместо – могуће је поставити неки други знак, нпр. +)
- Подржава различит број параметара за једну опцију (нпр. замишљена команда *-f/--funkcija* може да прима варијабилан број парамтера у зависности од саме намене функције)

Библиотека *Re*

Ова бибилиотека омогућава коришћење регуларних израза у програмском језику *Python*. Следи детаљније објашњење у следећем поглављу. [3]

Библиотека *Docx*

Наведеном библиотеком могуће је креирање и модификовање *.docx* датотека. [4]

Библиотека *Platform*

Користи се у добављању информација о подацима као што су хардвер, оперативни систем, верзија интерпретатора. Уз помоћ ове библиотеке се нпр. може добити информација о томе да ли је систем 64-битан или 32-битан или који је оперативни систем тренутно активан. [5]

Библиотека *Numpy*

Додаје функционалност за велике, мулти-димензионалне низове и матрице као и велику колекцију математичких функција које се могу применити на исте. Ова библиотека се користи и за мапирање објеката програмског језика *Python* у излазне датотеке који могу да чувају изведене податке за поновну употребу. [6]

Библиотека *PySpellChecker*

Представља библиотеку у оквиру програмског језика *Python* која пружа подршку у детектовању речи које су у тексту написане погрешно. За сваку препознату погрешно написану реч, наведена библиотека нуди предложену замену. У позадини деловања, користи се алгоритам Левенштајновог растојања. [7]

GhostScript

Слика 2: Лого *Ghostscript-a*

Представља алат који је базиран на интерпретеру за *PostScript(PS)* и *PortableDocumentFormat (PDF)* дескриптивне језике. Главна његова улога јесте у растеризацији односно рендеровању страница наведених формата, за приказ или штампање страница докумената као и конверзије између PS и PDF датотека. [8]

Наведени алат нуди доста функционалности које се могу наћи на њиховом званичном сајту. Две функционалности које су битне за софтверски алат који је тема овог рада су :

1. Конверзија *.ps* датотеке у *.pdf* формат.
2. Спајање више индивидуалних *.pdf* датотека у једну датотеку.

Paps

Наведени софтверски алат конвертује текстуалну датотеку *.txt* екстензије у *.ps* формат. Користи *UTF-8* кодирање које подржава све знакове и латиничног и ћириличног писма који се појављују у српском језику. [9]

LibreOffice Writer

У оквиру *LibreOffice* пакета, налази се такозвани *LibreOffice* W*riter* који представља пандан *MicrosoftOffice Word*-у. У овом алату је могуће отварати, модификовати и снимити датотеке које припадају *MicrosoftOffice* пакету (*.doc, .docx*). Из њега је могуће конвертовати и *.docx* датотеке у *.pdf* формат. Сам алат нуди могућност позивања те функционалности из командне линије односно терминала. [10]

Слика 3: Лого алата *LibreOffice Writer*

OfficeToPdf

Представља алат базиран на командној линији који конвертује *.docx* у *.pdf* формат. Многи алати који врше овакав вид конверзије јесу у ствари *desktop* апликације, док наведени алат нуди брзу и једноставну могућност конвертовања датотеке без додатних корака типа „*Save as…"*. [11]

Да би се алат користио, на оперативном систему *Windows* мора бити инсталирано:

- *.NET Framework 4*
- *Office* 2016, 2013, 2010 или 2007

Уколико корисник има инсталиран *Office* 2007, потребно је инсталирати и:

- *Visual Studio* 2010 алате за покретање *Office*
- 2007 *Microsoft Office Add-in*: Могућност снимања као *PDF* или *XML*

Појмови

Текстуална датотека (.txt)

```
Diplomski rad
tema:    Računarski alat za        jednostavnu obradu tekstualnih
datoteka baziran na komandnoj liniji.

Zadatak:
Имплементиратис рачунарски           алатр који се покреће из командне
линије и као улаз прима текстуалу датотексу екстензије txt и
docx.неопходно је направити помоћну наредбу -p, --pomoć која ће кориснику
дати кјасан преглед свих          могућности које наведении алат нуди.
након извршених жељених измена,излазна       датотека се може снимитио
под екстензијом коју има и улазна (измењена) датотека или пак у pdf
формату.
Резиме рада:
Уводни део рада почиње са упознавањем технологија       које су коришћене
у раду и које учествују у формирању крајње верзије рачунарског алата. пре
дела који је везан за сам алат, његово имплементирање и коришћење,
читалац ће се такође упознати и са појмовима као што су регуларни
изрази,Левенштајново растојање. након што се          читалац упозна са
технологијама и појмовима које је потребно знати,следи објашњење саме
логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење
везано за њено покретање.
Кључне речи:
Рачунарски алат,текстуална датотека, Pajton
ФУНКЦИОНАЛНОСТИ:

 Transformacija malih slova u velika gde god je to potrebno - početak
rečenice,posle tačke.
 Nakon svakog zareza,ukoliko nema razmaka,dodati ga kao i nakon tačke.
 Ukloniti duple ili višestruke razmake izmedu dve reči ili karaktera i
reči.
 Transformacija iz latinice u ćirilicu i obrnuto.
 Eksportovanje datoteke u pdf.
 Mogućnost spajanja više pdf datoteka u jednu pdf datoteku.
 Najbliže pronađena reč.
 Reči koje se najčešće pojavljuju u tekstu.
 Pronađi prvo pojavljivanje          unesene sekvence i       zameni
drugim unosom | Pronadi svako pojavljivanje unesene sekvence i zameni
drugim unosom.
```

Слика 4: Пример текстуалне датотеке

Представља тип датотеке која може да садржи само текст без икаквог специјалног формата као што је болдован, искошен, подвучен текст. Такође, не може садржати објекте као што су слике, табеле... Због своје једноставности, користе се за брзо складиштење текстуалних информација. Скуп знакова који се користе у српском језику припадају *UTF-8* кодирању.

Датотека са екстензијом *.docx*

Diplomski rad

tema: Računarski alat za jednostavnu obradu tekstualnih datoteka baziran na komandnoj liniji.

Zadatak:

Имплементиратис рачунарски алатр који се покреће из командне линије и као улаз прима текстуалу датотексу екстензије *txt* и *docx*.неопходно је направити помоћну наредбу - *p, --помоћ* која ће кориснику дати кјасан преглед свих могућности које наведении алат нуди. након извршених жељених измена,излазна датотека се може снимитио под екстензијом коју има и улазна (измењена) датотека или пак у *pdf* формату.

Резиме рада:

Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата. пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су **регуларни изрази**,**Левенштајново растојање**. након што се читалац упозна са технологијама и појмовима које је потребно знати,следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење везано за њено покретање.

Кључне речи:

Рачунарски алат,текстуална датотека, *Pajton*

ФУНКЦИОНАЛНОСТИ:

- Transformacija malih slova u velika gde god je to potrebno - početak rečenice,posle tačke.
- Nakon svakog zareza,ukoliko nema razmaka,dodati ga kao i nakon tačke.
- Ukloniti duple ili višestruke razmake između dve reči ili karaktera i reči.
- Transformacija iz latinice u ćirilicu i obrnuto.
- Eksportovanje datoteke u *pdf*.
- Mogućnost spajanja više pdf datoteka u jednu *pdf* datoteku.
- Najbliže pronađena reč.
- Reči koje se najčešće pojavljuju u tekstu.
- Pronađi prvo pojavljivanje unesene sekvence i zameni drugim unosom | Pronađi svako pojavljivanje unesene sekvence i zameni drugim unosom.

Слика 5: Пример датотеке са *.docx* екстензијом

Главна разлика у текстовном могућношћу складиштења између *.docx* и *.txt* јесте та што прво наведене датотеке могу стилизовати текст (болдовати га, искосити, подвући, обојити...). Различити делови текста могу имати различите величине фонта, припадати различитој фамилији слова.

Шта је *paragraph* а шта *run* у оквиру *.docx* датотеке

Ако се погледа документ изнад, видећемо да се сама *word* датотека састоји из више *paragraph*-а (пасуса) – где сваки притисак тастера *enter* формира нови пасус. У већини случајева, пасус се састоји из више такозваних *run*-ова – делови текста у оквиру једног пасуса који имају исти стил.

Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата. пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су **регуларни изрази**,**Левенштајново растојање**. након што се читалац упозна са технологијама и појмовима које је потребно знати,следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење везано за њено покретање.

Слика 6: пример једног пасуса

На слици је приказан неформатиран и неуређен текст који се састоји из пет различитих *run*-ова:

1. Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата. пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су
2. **регуларни изрази**
3. , (има различити стил од претходног *run*-а)
4. **Левенштајново растојање**
5. . након што се читалац упозна са технологијама и појмовима које је потребно знати,следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење везано за њено покретање.

Регуларни изрази

Представљају секвенцу знакова који дефиништу такозвани **образац претраге**. Такви обрасци се користе углавном од стране алгоритама за претрагу ниски за операције као што су „Пронађи“ и „Пронађи и замени“. Могу се користити и при валидацији улазних података када корисник треба да се улогује или да се региструје на одређени систем. [12]

Сваки знак у обрасцу претраге је или такозвани **метакзнак** (што значи да има посебно значење) или је **регуларни** знак који има обично тј. подразумевано значење. Нпр. образац претраге „a.a“ знак „a“ ће представљати управо то слово док ће знак „ .“ представљати било који знак у тексту у којем се врши претрага. Самим тим овај образац ће пронаћи речи односно секвенцу знакова као што су: ала, ада, ама, a1a, аца... Комбинацијом горе наведена два типа знакова, може се вршити претрага специфичне секвенце у некој текстуалној датотеци.

Операторе који се користе у регуларним изразима можемо поделити у следеће групе:

- Логичка операција *ИЛИ*

  Ова операција се обележава знаком „ | “ и као што и сам назив говори, наћи ће оно што је пре самог знака или оно што се налази након знака, уколико се једна од тих секвенци налази у самом тексту.

  Пример: школа|школе

- Груписање знакова

  Ознаке овог оператора се састоје из заграда „ () “. Оне дефинишу оквир и дефинишу предност других операција које се могу комбиновати уз овај оператор.

  Пример: школ(а|е) ће радити исту ствар као и пример изнад.

  За разлику од обичних заграда, угласте заграде „ [] “ омогућавају проналажење било ког знакова унутар истих.

  Пример: [школа] ће пронаћи било који од знакова ш, к, о, л, а. Угласте заграде су погодне када се траже специјалне секвенце као што је нпр. било који једноцифрени број – [0 – 9] цртица у оквиру угластих заграда специфицира опсег.

- Квантификатори

  Користе се након навођења жељене секвенце знакова или групе и означавају колико се они пута могу појављивати односно понављати. Листа квантификатора:

  - ? – дозвољава једно или ниједно појављивање претходно наведеног знака. Пример: главн?и - наћи ће секвенце: глави или главни
  - * - дозвољава више или ниједно појављивање претходно наведеног знака. Пример: главн*и – наћи ће секвенце: глави, главни, главнни, главннни итд.

- + - дозвољава једно или више појављивања претходно наведеног знака. Пример: главн+и – наћи ће секвенце: главни, главнни, главннни итд.

- *Wildcard*-е

Сви знакови из ове групе оператора замењују одређене секвенце знакова. Најпознатији метакарактери су:

- \d – мења било коју цифру у распону од 0-9.
- \w – мења било који алфанумерички знак
- \W – мења било који знак који не припада групи алфанумеричких знакова ($#%)
- . – мења било који знак
- ^ - тражи унети образац на почетку секвенце. Када се користи у оквиру угластих заграда, представља свој опозит.
- $ - тражи унети образац на крају секвенце. Када се користи у оквиру угластих заграда, представља свој опозит.

Пример:

^[a-zA-Z]\w{8,14}$

Пример регуларног излаза за валидацију шифре

Чест је случај да се при регистрацији новог корисника захтева од самог корисника да испоштује одређене кораке приликом регистрације. Обично су ти захтеви да:

- Први знак мора бити слово (велико или мало)
- Остали знакови могу бити слова, бројеви или доња црта
- Обично величина шифре мора бити одређен број знакова (у овом примеру је минималан број знакова за шифру 8 а максималан 14)

Левенштајново растојање

У информатичкој теорији, лингвистици и рачунарству, **Левенштајново растојање** представља метрику ниски за мерење различитости између две секвенце знакова. Информативно гледано, Левенштајново растојање између две речи представља минимални број потребних промена појединачних знакова (убацивање новог, брисање или замена постојећег) да би се једна секвенца трансформисала у другу. Име је добила по математичару из Совјетског Савеза, Владимиру Левенштајну, који ју је патентирао 1965. године. [13]

У неким литературама, Левенштајново растојање се може навести и као **растојање уређивања** (edit distance) али се тај израз односи на ширу фамилију метрика рачунања дистанци.

Дефиниција

Левенштајново растојање између ниски *a* и *b* (дужине |*a*| и |*b*| респективно) је дата изразом $\boldsymbol{lev_{a,b}(|a|, |b|)}$ где је:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \,!= b_j)} \end{cases} & \text{у супротном} \end{cases}$$

Где је $1_{(ai \,!= bj)}$ индикаторска функција која је једнака 0 када је (a_i = b_j) а у супротном је једнака 1, $lev_{a,b}(i,j)$ растојање између првих „i“ знакова ниске *a* и првих „j“ знакова ниске *b*.

Примећује се да први елемент у одељку *min* одговара брисању знакова (из ниске *a* у *b*), други уметању знакова а трећи одговара провери једнакости два знака.

Пример:

Левенштајново растојање између речи „плесати“ и „јести“ је 3, с обзиром да се морају извршити три промене како би се прва реч трансформисала у другу:

1. плесати -> јлесати (замена слова п у слово ј)
2. јлесати -> јесати (брисање слова л)
3. јесати -> јести (брисање слова а)

Горња и доња гранична вредност

Левенштајново растојање има неколико ограничења. Та ограничења су:

- Вредност растојања минимално је једнака разлици величина две ниске које се упоређују.
- Максимална вредност растојања представља дужину веће од две ниске које се упоређују.
- Нула је ако и само ако су две ниске потпуно исте.
- Ако је величина ниски иста, Хамингово растојање је горња граница. [14]

  Пример: мраз крак -> Хамингово растојање између ове две речи је 3 јер је се разликују знакови на позицијама 1. 3. и 4. док је Левенштајново растојање у овом случају 2:

  - мраз -> краз
  - краз -> крак
- Вредност растојања између две ниске није већа од суме њихових Левенштајнових растојања од треће ниске.

  Последње тврђење се базира на теореми о **неједнакости троуглова** која каже да било која страница троугла није већа од збира дужина преостале две странице у троуглу. [15]

Примена

У апроксимативном тражењу ниски, циљ је наћи исправне солуције за ниске у великом тексту, у ситуацијама где се очекује мали број измена. Предложене измене тј. исправне солуције за одређену ниску могу доћи нпр. из речника. Тражење исправне солуције за веће ниске представља сложенији посао јер у таквим нискама постоји већи број могућих комбинација операција брисања, уметања или додавања новог знака.

Апликација овог алгоритма има широки спектар коришћења али се највише користи у:

1. Системима за проверу правописа (*spell check*)
2. Корекционим системима за оптичко препознавање знакова
3. Софтверима у којима асистира при природној транслацији језика који се базирају на транслационој меморији.

Функционалности

Скрипта *diplomski_rad.py*

Прикупљање аргумената са командне линије, врши се уз помоћ скрипте *diplomski_rad.py*. У зависности од унетих аргумената, извршиће се одговарајућа функција. У оквиру парсера, постоје три групе аргумената:

- *pomoć* – позивом овог аргумента, врши се приказ свих команди које наведени алат нуди уз објашњење шта који аргумент ради.
- *flegovi* – они евентуално допуњују позив функцији и пружају функционалности као што су:
    - *-i/--ignoriši* - позивом овог флега се велика и мала слова третирају исто. Примењује се евентуално на функцију која излистава најфреквентније речи у тексту или приликом замене одређене секвенце у тексту, новом секвенцом.
    - *-z/--zameni_sve* – позивом наведеног флега се мења свако појављивање наведене секвенце новом секвенцом у текстуалној датотеци и може се применити само на ту функционалност у скрипти.
    - *-o/--izlaz* – може се применити на функције које врше модификацију улазне датотеке, стварајући нову, излазну датотеку.
- *funkcije* – софтверски алат нуди девет функционалности:
    - *-1/--jedan* – тансформација малих слова у велика где год је то потребно – почетак реченице, након тачке.
    - *-2/--dva* – додаје размак након сваке запете или тачке и речи уколико између њих не постоји размак.
    - *-3/--tri* – уклања дупле или вишеструке размаке између две речи.
    - *-4/--četiri* – трансформише латиничне речи у ћирилицу и обрнуто у зависности од потребе.
    - *-5/--pet* – конвертује *.txt* или *.docx* датотеку у *.pdf* формат.
    - *-6/--šest* – спаја више *.pdf* датотека у једну *.pdf* датотеку.
    - *-7/--sedam* – проверава да ли постоји нека реч која је погрешно написана.
    - *-8/--osam* – штампа речи које се најчешће понављају.
    - *-9/--devet* – уз помоћ ове функције могуће је заменити секвенцу у тексту новом секвенцом на само једном месту или свуда где се та секвенца појављује.

Након прегледа могућих аргумената, потребно је поставити неколико ограничења:

- Уколико се при позиву аргумента *-p/--pomoć* унесе било шта друго, штампаће се информације које пружају преглед о сваком понуђеном аргументу
- Уколико се било која функција (осим функције *-6/--šest*) позове без навођења улазне датотеке, штампаће се информација о потребном уносу истог.
- Флег *-o/--izlaz* се може применити само на функције које модификују улазну датотеку. У супротном се штампа одговарајућа порука.
- Флег *-z/--zameni_sve* се може применити само на функцију девет. У супротном се штампа одговарајућа порука.
- Флег *-i/--ignoriši* се може применити само на функцију девет и осам. У супротном се штампа одговарајућа порука.
- Уколико се унесе непознати аргумент а да притом није наведен аргумент који штампа помоћ, исписаће се одговарајућа порука.

У зависности од позване функције, скрипта *diplomski_rad.py* ће се преусмерити на позивање исте у скрипти *funkcije.py*.

Скрипта *funkcije.py*

Функција један

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос.

Датотека *.txt*

Пошто текст у овом типу датотека не подржава стилизовање, читав текст се може учитати одједном и сместити у одговарајућу променљиву. Након тога, следи провера првог знака – уколико је написано слово мало, врши се његово трансформисање у велико слово. Када се наиђе на тачку приликом испитивања преосталих „дужина датотеке - 1“ знакова, прво се врши провера да ли се испред тачке налази број (у том случају то значи да се ради о редном броју) или нека скраћеница (скраћенице се налазе у скрипти *recnik.py*). У обе ситуације не треба извршити капитализацију речи која се налази иза тачке. Остатак знакова се уписује редом у листу знакова који ће се касније наћи у датотеци.

Датотека *.docx*

Овде је ситуација за нијансу комплекснија од претходно наведене. Као што је објашњено у секцији ***Појмови***, овакав тип датотеке се састоји од више пасуса а сваки пасус углавном од више такозваних *run*-ова. Учитавање података се врши тако што се испитује пасус по пасус а у оквиру пасуса *run* по *run*. Сваки *run* може да се заврши тачком што би значило да треба проверити почетак наредног *run*-а. Провера се врши само уколико та реч није скраћеница или нумеричка вредност. У супротном ће се капитализовати прва реч у следећем *run*-у. Приликом уписивања знакова сваког *run*-а, потребно је водити о стилу који текст има и потребно га је сачувати.

Након испитаних свих знакова, врши се спајање листе у ниску и снимање измењене датотеке:

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функција два

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос.

Датотека *.txt*

Учитани подаци из датотеке се складиште у виду ниске. Пошто су у програмском језику *Python* ниске непроменљиви типови података, потребно је да те податке складиштимо у листу, знак по знак. Након тога се врши итерација кроз листу и када год се наиђе на знак који представља запету или тачку, врши се испитивање следећег знака у листи. Уколико тај следећи знак није размак, додаје се на позицију након запете/тачке.

Датотека *.docx*

Логика је слична као у претходно наведеној ситуацији с тим што је могући сценарио да се *run* у оквиру пасуса завршава запетом односно тачком. У тој ситуацији потребно је проверити почетак наредног *run*-а у оквиру истог пасуса. Уколико почетак наредног пасуса није размак, то значи да између запете/тачке не постоји размак и треба га убацити.

Након испитаних свих знакова, врши се спајање листе у ниску и снимање измењене датотеке:

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функција три

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос.

Датотека *.txt*

У решавању проблема виштеструких размака, користе се регуларни изрази (детаљно описани у одељку *Инструкције*. Програмски језик *Python* има библиотеку по имену *Re* која садржи неколико функција па између осталог и функцију за замену одређене секвенце у оквиру ниске. Образац регуларног израза за замену је ‘ +’ док је образац којом треба заменити секвенцу ‘ ‘, што значи да уколико поред једног размака постоји још један или више њих, обрисаће се сви размаци осим једног.

Датотека *.docx*

Размаци увек припадају једном *run*-у па тако не постоји проблем да се одређена доза размака нађе у једном а затим и у другом *run*-у. Самим тим логика је иста као у случају решавања проблема везаног за датотеке са *.txt* екстензијом.

Након извршене модификације, снимање измењене датотеке се може обавити :

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функција четири

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос. Као параметар се наводи и жељено писмо у које је потребно извршити конверзију.

Датотека *.txt*

Приликом трансформисања речи из ћирилице у латиницу, ситуација је кристално чиста. Сваки знак у ћирилици има одговарајући еквивалент у латиничном писму и алгоритам тако и ради. Када год наиђе на ћирилични знак (а потребно је трансформисати текст у латиницу), из скрипте *recnik.py* извлачи се латинични еквивалент ћириличног знака. Приликом трансформације латиничне речи у ћириличну ствар је мало комплекснија. Латинично писмо не функционише по принципу латиничног – једно слово један знак. Сам алгоритам функционише по следећем принципу – прво се сваки знак учитаног текста појединачно уписује у листу. Након тога се врши итерација кроз новноасталу лису и врши спајање знакова у речи. Сви остали знакови (тачке, запете, знаци навода итд.) се налазе самостално у листи.

У српском језику постоје речи које секвенцу слова као што је нпр. „nj“ не јотују већ управо тако записују и у ћириличном писму (реч *konjugacija* треба ћирилично да се напише *конјугација* а не *коњугација*). Самим тим је потребно на неки начин извршити проверу када треба извршити јотовање а када не. У скрипти *recnik.npy* се налазе латиничне речи које садрже свој одговарајући ћирилични еквивалент. Речи из оба писма су преузета из речника који се налази у *LibreOffice* алату.

На тај начин алгоритамски се прво проверава да ли је реч у листи пронађена у речнику и ако јесте, замениће га својим ћириличним еквивалентом. У супротном ће се вршити итерација кроз листу и уписивати знак по знак са одређеним изузетком:

- Уколико се знакови „nj“, „lj“ или „dž“ налазе један поред другог, пре појединачног уписивања знакова, врши се њихово спајање у једну ниску а затим упис у листу у којој се на крају налази формирана реч. Затим у следећој итерацији знак „j“ или „ž“ не треба уписивати јер су се спојили са својим претходницима.

Новонастала реч се спаја у ниску и уписује се на позицији где се некада налазила иста та латинична реч.

Датотека *.docx*

Логика је потпуно иста као и у претходном случају са тим што се мора водити рачуна о речима у самом *run*-у. Може се десити да знакови једне речи припадају различитим *run*-овима што значи да се може десити да се латинична реч не конвертује скроз тачно у свој еквивалент. Размотримо ситуацију у којој део *коњ* припада једном *run*-у а део *укцција* другом *run*-у. Пошто алгоритам функционише тако што испитује *run* по *run*, он би ове две секвенце раздвојио и спојена верзија ове две наизглед одвојене речи би била „коњукција“ (исправно би било конјукција). Самим тим се пре испитивања сваког *run*-а испитује завршетак претходног – уколико се претходни завршио словним знаком, а тренугни *run* почиње словним знаком, сигурно је у питању једна реч и треба та два *run*-а посматрати као један те исти.

Након испитаних свих знакова односно речи, врши се спајање листе у ниску и снимање измењене датотеке:

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функција пет

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос.

Уколико датотека прође проверу валидације и уколико је екстензије *.txt*, текстуална датотека ће добити и свој *.pdf* еквивалент. Прво се врши конверзија текстуалне датотеке у *.ps (PostScript)* формат а након тога се позива функција *GhostScript* алата за конверзију .ps датотеке у *.pdf*.

Уколико је датотека пак *.docx* екстензије, конверзија улазне датотеке у *.pdf* формат зависи од оперативног система. Пошто је природно да *Linux* оперативни системи имају инсталиран *LibreOffice* (углавном овај алат постоји самом инсталацијом *Linux*-а), а *Windows* оперативни систем *MicrosoftOffice*, самим тим се наведени алати користе на оперативним системима где подразумевано постоје.

Функција шест

Прима одређену дозу аргумената са командне линије. Прво се врши провера да ли су првих *n - 1* (улазне датотеке које треба спојити) датотека у ствари *.pdf* датотеке. Уколико јесу врши се избор имена излазне датотеке које такође морају бити *.pdf* формата. Последња *n*-та датотека представља излазну датотеку у којој ће се складиштити претходно наведене спојене датотеке. Спајање се врши уз помоћ алата *GhostScript*. Због разлике у покретању самог алата на различитим оперативним системима, раздвојене су команде покретања за *Linux* и *Windows* оперативне системе.

Функција седам

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос. Као параметар се прослеђује број који представља Левенштајнову дистанцу. Користи функције два и један као услужне функције. Препоруке за секвенце знакова за које се претпостави да су погрешно написане ће се добијати из речника *sr_full.txt* у којем се налази скоро 2 000 000 речи, што латиничних што ћириличних.

Датотека *.txt*

Пошто је за ову функционалност потребно посматрати само речи у тексту, све остало из истог се брише тако што ће се позвати прво *split* метода у оквиру програмског језика *Python*, која ће ниску поделити по размацима које је нашао и сместити у листу података. На тај начин може да се деси да се уз реч нађе додатни знак као што је тачка, запета или две тачке, па је потребно те знакове и уклонити. Регуларним изразом се може врло лако решити наведени проблем. Образац претраге је ‘\W+’ који ће наћи све знакове који нису слова или бројеви и обрисати пошто је образац замене ‘’ (празна ниска). Таква новонастала листа може садржати празне ниске које је такође потребно уклонити.

У оквиру ове функције се користи библиотека *pyspellchecker*. Навођењем функције *unknow* наведене библиотеке, налазе се све речи у листи за које се сматра да су погрешно написане. Након тога се итеративно приказује реч по реч за које се сматра да су погрешно написане. За сваку од тих речи корисник ће бити упитан да ли жели да измени одговарајућу секвенцу знакова новом, препорученом вредношћу. Пошто је рачунање Левенштајнове дистанце код речи сачињених од више знакова временски захтевно, за њих ће се увек користити растојање један. Методом *candidates* се из поменуте листе *sr_full.txt* налази одговарајућа препорука. Ради лакшег коришћења, уз сваку препоруку је потребно додати и индекс који је потребно унети у конзолу уколико се изабере баш та, жељена реч. У сваком тренутку корисник може да прекине рад програма уписивањем речи *izlaz*.

Датотека *.docx*

Слично ситуацији у функцији четири где је објашњено да је могуће да се једна реч може наћи у два различита *run*-а, да би исправка секвенце радила добро, мора се водити рачуна о томе да се две одвојене порције једне речи нађу у оквиру истог *run*-а. Логички је све исто као у коду који се односи на рад са *.txt* датотекама.

Након испитаних свих речи, снимање измењене датотеке могуће је:

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функција осам

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос. Као параметар се наводи и број којим ће се приказати толико наведних најфреквентнијих речи у тексту и флег којим ће се означити да ли је битно разликовати мала и велика слова у речи или не.

Датотека *.txt*

Пошто је за ову функционалност потребно посматрати само речи у тексту, све остало из истог се брише тако што ће се позвати прво *split* метода у оквиру програмског језика *Python*, која ће ниску поделити по размацима које је нашао у листу података. На тај начин може да се деси да се уз реч нађе додатни знак као што је тачка, запета или две тачке, па је потребно те знакове и уклонити. Регуларним изразом се може врло лако решити наведени проблем. Образац претраге је '\W+' који ће наћи све знакове који нису слова или бројеви и заменити обрисати пошто је образац замене '' (празна ниска). Уколико је унет флег којим се мала и велика слова занемарују, нпр. реч *Кућа* и *кућа* ће се посматрати исто. Ако се деси да корисник унесе већи број од броја пронађених речи у тексту, врши се приказ свих пронађених речи у тексту и исписује се одговарајућа порука кориснику о прекорачењу броја речи које је могуће пронаћи.

Датотека *.docx*

Наведена функција је једна од три које не модификују улазну датотеку па самим тим није потребно водити рачуна о стилу текста. Логика је потпуно иста с тим што се уместо учитавања целокупног текста, врши учитавање пасуса по редоследу појављивања у тексту.

Функција девет

На самом почетку, врши се провера уноса датотеке. Уколико је унета датотека која не постоји, штампаће се одговарајућа порука. Такође, уколико је унета датотека која нема екстензију *.docx* iili *.txt*, приказаће се порука која упозорава корисника који су то дозвољени формати за унос. Као параметре ова функција прима секвенцу коју је потребно заменити новом секвенцом која се уноси као наредни параметар. Опционо се наводе и два флега – флег којим ће се заменити свако појављивање старе секвенце новом секвенцом у тексту као и флег којим ће се мала и велика слова приликом тражења секвенце посматрати исто.

Датотека *.txt*

Пошто је за ову функционалност потребно посматрати само речи у тексту, све остало из истог се брише тако што ће се позвати прво *split* метода у оквиру програмског језика *Python*, која ће ниску поделити по размацима које је нашао, у листу података. На тај начин може да се деси да се уз реч нађе додатни знак као што је тачка, запета или две тачке, па је потребно те знакове и уклонити. Регуларним изразом се може врло лако решити наведени проблем. Образац претраге је '\W+' који ће наћи све знакове који нису слова или бројеви и заменити обрисати пошто је образац замене ''

(празна ниска). Пошто постоје два флега која се могу специфицирати на наведену функционалност, постоји 2^2 могућих сценарија:

1. Да су унети и флег замене сваког појављивања секвенце у тексту и флег којим се игноришу мала и велика слова.

   Пример: секвенца знакова коју желимо да заменимо је „факултет“ у тексту су пронађене речи „Факултет факултет фАкултет факултет“. Све четири речи ће бити замењене новоизабраном секвенцом знакова.

2. Да је унет само флег којим се игноришу мала и велика слова.

   Пример: секвенца знакова коју желимо да заменимо је „факултет“ у тексту су пронађене речи „Факултет факултет фАкултет факултет“. Замениће се само реч „Факултет“ новоизабраном секвенцом знакова.

3. Да је унет само флег замене сваког појављивања секвенце у тексту.

   Пример: секвенца знакова коју желимо да заменимо је „факултет“ у тексту су пронађене речи „Факултет факултет фАкултет факултет“. Замениће се свако појављивање речи „факултет“ новоизабраном секвенцом знакова.

4. Да није унет ни један од наведених флегова.

   Пример: секвенца знакова коју желимо да заменимо је „факултет“ у тексту су пронађене речи „Факултет факултет фАкултет факултет“. Замениће се прво појављивање речи „факултет“ новоизабраном секвенцом знакова.

Датотека *.docx*

Као и у функцијама четири и седам тако и овде може доћи до појављивања да се два различита дела једне речи нађу у два различита *run*-а па тај проблем треба решити. Такође, пошто се овде проверава пасус по пасус а у оквиру сваког пасуса *run* по *run*, потребно је специфицирати флег провере који ће вршити проверу да ли је већ дошло до замене речи уколико није специфициран флег замене. Уколико је дошло до једне (прве) промене у тексту, онда није више потребно модификовати текст. Остатак алгоритма је идентичан делу везаном за *.txt* датотеке.

Након испитаних свих речи, снимање измењене датотеке могуће је:

- Под истим именом као и улазна датотека (преснимити улазну датотеку)
- Под различитим именом

Функције својства *run*-а, тип датотеке и име датотеке

Наведене функције се користе као услужне функције у другим функцијама. Прва наведена функција садржи одређене стилове које треба применити на одговарајући *run* као што су болдован, подвучен, искошен, обојен текст итд. Друга функција очитава екстензију унете датотеке тако што налази последње наведену тачку у оквиру назива датотеке и од те позиције враћа завршну секвенцу знакова. Последње наведена функција ради супротно од претходно наведене – памти позицију последње пронађене тачке у називу датотеке и враћа све што се налази пре исте.

Скрипта *recnik.py*

Као што су неке функције услужне, тако се и ова скрипта може окарактерисати као услужна. У оквиру ње се налазе два речника :

- Први речник садржи слова латиничног и ћириличног писма тако да је кључ латинично слово а вредност под тим кључем његов ћирилични еквивалент.
- Други речник садржи списак од неколико скраћеница у српском језику на латиници и ћирилици.

Приликом позива неке од функционалности које захтевају рад са наведеним речницима, врши се управо приступ овој скрипти (нпр. трансформација из латинице у ћирилицу и обрнуто).

Преостале функције ове скрипте су искоришћене за креирање датотеке под именом *recnik.npy*. Вредности у овом речнику су сачуване у облику речника програмског језика *Python* – латинична реч представља кључ а њен ћирилични еквивалент вредност под тим кључем.

Покретање и коришћење алата

Као што је још у самом наслову наведено, покретање самог алата се врши из командне линије навођењем жељене функционалности коју је потребно испунити. Прва веома значајна команда је:

```
python diplomski_rad.py -p
```

Слика 7: Покретање опције за приказивање помоћи

Покретањем ове команде, исписаће се следеће:

```
usage: diplomski_rad.py [-p] [-i] [-z] [-o IZLAZ] [-f FAJL] [-1] [-2] [-3]
                        [-4 ČETIRI] [-5] [-6 [ŠEST [ŠEST ...]]] [-7 SEDAM]
                        [-8 OSAM] [-9 DEVET DEVET]

Softverski alat kojim se ulazni tekstualni fajl ekstenzije .txt/.docx
transformiše u željeni izlaz

pomoć:
  -p, --pomoć           Pruža objašnjenje svake funkcije

flegovi:
  -i, --ignoriši        Kada se pozove kao argument, velika i mala slova će se
                        posmatrati isto. "\Može se upariti sa funkcijama
                        -8/--osam, -9/--devet
  -z, --zameni_sve      Omogućava da se svako/prvo pojavljivanje specificirane
                        reči u tekstu zameni novom sekvencom. Ukoliko se
                        navede kao argument, zameniće se svako pojavljivanje
                        navedene sekvence - u suprotnom će se zameniti samo
                        prva pronađena sekvenca. Može se upariti samo sa
                        funkcijom -9/--devet
  -o IZLAZ, --izlaz IZLAZ
                        Izlaz će biti novi fajl. Potrebno je navesti ime
                        izlaznog fajla bez navođenja ekstenzije jer će
                        ekstenzija biti jednaka ekstenziji ulaznog fajla. Može
                        se primeniti na svaku funkciju koja vrši modifikaciju
                        fajla - ne može se primeniti na funkcije -5/--pet,
                        -6/--šest, -8/--osam
```

Слика 8: Приказ првог од два дела опције помоћ

Први део команде помоћ исписује информације о томе која функција односно флег, захтева колико прослеђених аргумената. Поред сваког флега налази се објашњење о томе шта сам флег треба да уради када се примени на неку од функција.

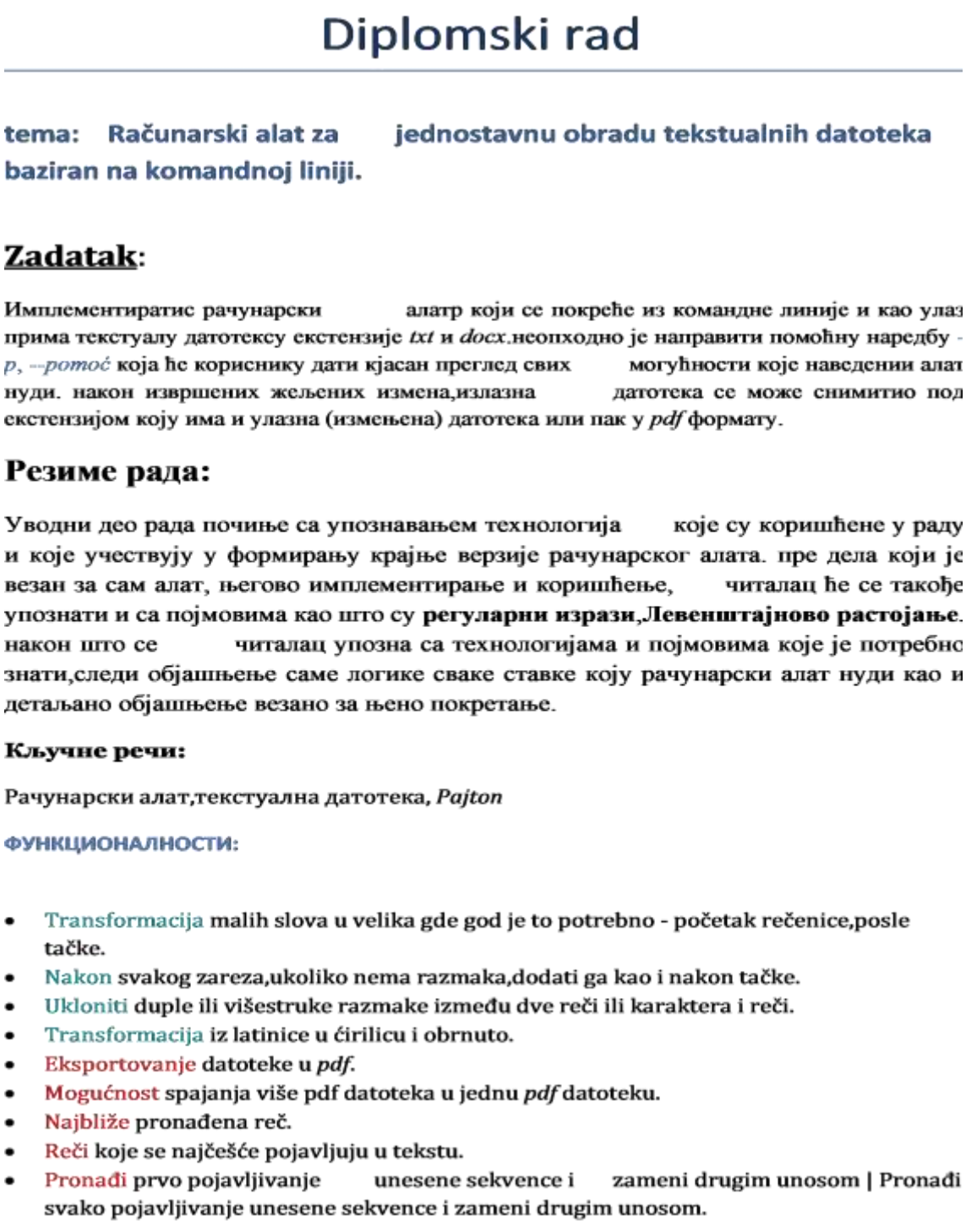
```
funkcije:
  -f FAJL, --fajl FAJL  Naziv ulaznog fajla
  -1, --jedan           Transformacija malih slova u velika gde god je to
                        potrebno - početak rečenice, posle tačke
  -2, --dva             Nakon svakog zareza, ukoliko nema razmaka, dodati ga
                        kao i nakon tačke
  -3, --tri             Ukloniti duple ili višestruke razmake između dve reči
  -4 ČETIRI, --četiri ČETIRI
                        Transformacija iz latinice u ćirilicu i obrnuto. Kao
                        argument se navodi pismo u koje je potrebno izvršiti
                        konverziju teksta - l/latinica za latinično pismo,
                        ć/ćirilica za ćirilično pismo
  -5, --pet             Eksportovanje fajla u pdf
  -6 [ŠEST [ŠEST ...]], --šest [ŠEST [ŠEST ...]]
                        Mogućnost spajanja više pdf fajlova u jedan pdf fajl -
                        poslednje navedeni pdf fajl predstavlja izlaz
  -7 SEDAM, --sedam SEDAM
                        Najbliže pronađena reč. Kao argument se navodi
                        distanca (broj slova koja se razlikuju između pogrešno
                        napisane reči i nove, predložene reči) na osnovu koje
                        će se tražiti odgovarajuće reči za pogrešno unesenu
                        reč.
  -8 OSAM, --osam OSAM  Reči koje se najčešće pojavljuju u tekstu - kao
                        paramter se unosi broj 'n' kojim se ispisuje prvih 'n'
                        najčešće upotrebljenih reči u tekstualnom fajlu. Ovu
                        funkcionalnost moguće je kombinovati uz fleg
                        -i/--ignoriši
  -9 DEVET DEVET, --devet DEVET DEVET
                        Pronađi prvo pojavljivanje unesene sekvence i zameni
                        drugim unosom | Pronađi svako pojavljivanje unesene
                        sekvence i zameni drugim unosom. Navode se dva
                        parametra. Prvi treba da bude sekvenca koju želimo da
                        zamenimo, zatim se unosi nova sekvenca kojom želimo da
                        zamenimo navedenu. Ukoliko je potrebno zameniti svako
                        pojavljivanje sekvence, unosi se fleg -z. Ukoliko nije
                        bitno da li je navedena sekvenca koju je potrebno
                        zameniti sačinjena od malih odnosno velikih slova,
                        unosi se i fleg -i
```

Слика 9: Приказ другог од два дела опције помоћ

Корисник на овај начин може лако да савлада коришћење самог алата јер је поред сваке од функционалности наведено шта та функционалност ради и са којим флегом може да се упари.

Примена функционалности

Приказ свих функционалности биће приказана на примеру следеће *.docx* датотеке:



Diplomski rad

tema: Računarski alat za jednostavnu obradu tekstualnih datoteka baziran na komandnoj liniji.

Zadatak:

Имплементиратис рачунарски алатр који се покреће из командне линије и као улаз прима текстуалу датотексу екстензије *txt* и *docx*.неопходно је направити помоћну наредбу -*p*, *--pomoć* која ће кориснику дати кјасан преглед свих могућности које наведении алат нуди. након извршених жељених измена,излазна датотека се може снимитио под екстензијом коју има и улазна (измењена) датотека или пак у *pdf* формату.

Резиме рада:

Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата. пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су **регуларни изрази**,**Левенштајново растојање**. након што се читалац упозна са технологијама и појмовима које је потребно знати,следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење везано за њено покретање.

Кључне речи:

Рачунарски алат,текстуална датотека, *Pajton*

ФУНКЦИОНАЛНОСТИ:

- Transformacija malih slova u velika gde god je to potrebno - početak rečenice,posle tačke.
- Nakon svakog zareza,ukoliko nema razmaka,dodati ga kao i nakon tačke.
- Ukloniti duple ili višestruke razmake između dve reči ili karaktera i reči.
- Transformacija iz latinice u ćirilicu i obrnuto.
- Eksportovanje datoteke u *pdf*.
- Mogućnost spajanja više pdf datoteka u jednu *pdf* datoteku.
- Najbliže pronađena reč.
- Reči koje se najčešće pojavljuju u tekstu.
- Pronađi prvo pojavljivanje unesene sekvence i zameni drugim unosom | Pronađi svako pojavljivanje unesene sekvence i zameni drugim unosom.

Слика 10: Приказ неуређеног текста

Покретањем функције један на дату датотекту, модификоваће се иста тако што ће сваку реч коју је потребно у тексту капитализовати (почетно слово велико).

```
python diplomski_rad.py -f demo.docx -1
```

Слика 11: Приказ покретања прве команде

Покретањем функције два на дату датотеку, тамо где не постоји размак између запете односно тачке и речи, размак се додаје.

```
python diplomski_rad.py -f demo.docx -2
```

Слика 12: Приказ покретања друге команде

Након покретања треће функције, сваки виштеструки размаци између два знака ће се трансформисати у само један размак.

```
python diplomski_rad.py -f demo.docx -3
```

Слика 13: Приказ покретања треће команде

Четвртом командом се специфицира жељено писмо у које је потребно извршити конвертовање. На ову као и на горе наведене три функције се може применити флег -o који ствара нову датотеку. Следећим паром команди ће тренутна датотека постати у потпуности латинична док ће се створити и нова ћирилична датотека.

```
python diplomski_rad.py -f demo.docx -4 L
```

Слика 14: Приказ покретања четврте команде

```
python diplomski_rad.py -f demo.docx -4 ć -o demo_ćirilica
```

Слика 15: Приказ покретања четврте команде која ствара нову датотека

Тренутни изглед датотеке *demo.docx*:

Diplomski rad

Tema: Računarski alat za jednostavnu obradu tekstualnih datoteka baziran na komandnoj liniji.

Zadatak:

Implementiratis računarski alatr koji se pokreće iz komandne linije i kao ulaz prima tekstualu datoteksu ekstenzije *txt* i *docx*. Neophodno je napraviti pomoćnu naredbu *-p, --pomoć* koja će korisniku dati kjasan pregled svih mogućnosti koje navedenii alat nudi. Nakon izvršenih željenih izmena, izlazna datoteka se može snimitio pod ekstenzijom koju ima i ulazna (izmenjena) datoteka ili pak u *pdf* formatu.

Rezime rada:

Uvodni deo rada počinje sa upoznavanjem tehnologija koje su korišćene u radu i koje učestvuju u formiranju krajnje verzije računarskog alata. Pre dela koji je vezan za sam alat, njegovo implementiranje i korišćenje, čitalac će se takođe upoznati i sa pojmovima kao što su **regularni izrazi**, **Levenštajnovo rastojanje**. Nakon što se čitalac upozna sa tehnologijama i pojmovima koje je potrebno znati, sledi objašnjenje same logike svake stavke koju računarski alat nudi kao i detaljano objašnjenje vezano za njeno pokretanje.

Ključne reči:

Računarski alat, tekstualna datoteka, *Pajton*

FUNKCIONALNOSTI:

- Transformacija malih slova u velika gde god je to potrebno - početak rečenice, posle tačke.
- Nakon svakog zareza, ukoliko nema razmaka, dodati ga kao i nakon tačke.
- Ukloniti duple ili višestruke razmake između dve reči ili karaktera i reči.
- Transformacija iz latinice u ćirilicu i obrnuto.
- Eksportovanje datoteke u *pdf*.
- Mogućnost spajanja više pdf datoteka u jednu *pdf* datoteku.
- Najbliže pronađena reč.
- Reči koje se najčešće pojavljuju u tekstu.
- Pronađi prvo pojavljivanje unesene sekvence i zameni drugim unosom | Pronađi svako pojavljivanje unesene sekvence i zameni drugim unosom.

Слика 16: Приказ делимично уређене латиничне датотеке

Ћирилична датотека *demo_ćirilica.docx* изгледа:

Дипломски рад

Тема: Рачунарски алат за једноставну обраду текстуалних датотека базиран на командној линији.

Задатак:

Имплементиратис рачунарски алатр који се покреће из командне линије и као улаз прима текстуалу датотексу екстензије *тхт* и *доцх*. Неопходно је направити помоћну наредбу *-п, --помоћ* која ће кориснику дати кјасан преглед свих могућности које наведении алат нуди. Након извршених жељених измена, излазна датотека се може снимитио под екстензијом коју има и улазна (измењена) датотека или пак у *пдф* формату.

Резиме рада:

Уводни део рада почиње са упознавањем технологија које су коришћене у раду и које учествују у формирању крајње верзије рачунарског алата. Пре дела који је везан за сам алат, његово имплементирање и коришћење, читалац ће се такође упознати и са појмовима као што су **регуларни изрази**, **Левенштајново растојање**. Након што се читалац упозна са технологијама и појмовима које је потребно знати, следи објашњење саме логике сваке ставке коју рачунарски алат нуди као и детаљано објашњење везано за њено покретање.

Кључне речи:

Рачунарски алат, текстуална датотека, *Пајтон*

ФУНКЦИОНАЛНОСТИ:

- Трансформација малих слова у велика где год је то потребно - почетак реченице, после тачке.
- Након сваког зареза, уколико нема размака, додати га као и након тачке.
- Уклонити дупле или вишеструке размаке између две речи или карактера и речи.
- Трансформација из латинице у ћирилицу и обрнуто.
- Експортовање датотеке у *пдф*.
- Могућност спајања више пдф датотека у једну *пдф* датотеку.
- Најближе пронађена реч.
- Речи које се најчешће појављују у тексту.
- Пронађи прво појављивање унесене секвенце и замени другим уносом | Пронађи свако појављивање унесене секвенце и замени другим уносом.

Слика 17: Ћирилична верзија датотеке *demo.docx*

Тренутна датотека се можемо снимити у *.pdf* формату позивом функције пет:

```
python diplomski_rad.py -f demo.docx -5
```

Слика 18: Приказ покретања пете команде

Такође, могуће је повезати нпр. латиничну и ћириличну *.pdf* верзију у једну датотеку:

```
python diplomski_rad.py -6 demo.pdf demo_ćirilica.pdf spojeni.pdf
```

Слика 19: Приказ покретања шесте команде

Као што се може приметити са претходних слика, постоје речи које су неправилно написане и које морају бити исправљене. Покретањем функције седам се врши исправка. Кориснику ће се интерактивно приказивати речи које су препознате као погрешно написане:

```
python diplomski_rad.py -f demo.docx -7 1
```

Слика 20: Приказ покретања седме команде

```
Реч alatr је можда погрешно написана. Желите ли да реч "alatr" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč alatr: ['alatu -> indeks: 1', 'alati -> indeks: 2', 'alar -> indeks: 3', 'latr -> indeks: 4', 'alatir -> indeks: 5', 'alate -> indeks: 6', 'alat -> ind
eks: 7', 'alata -> indeks: 8', 'alator -> indeks: 9', 'altar -> indeks: 10']
Unesite indeks: 7
Реч detaljano је можда погрешно написана. Желите ли да реч "detaljano" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč detaljano: ['detaljanu -> indeks: 1', 'detaljana -> indeks: 2', 'detljano -> indeks: 3', 'detaljno -> indeks: 4', 'detaljan -> indeks: 5']
Unesite indeks: 4
Реч ievinstajnova је можда погрешно написана. Желите ли да реч "ievinstajnova" замените неком другом секвенцом карактера?
da|ne|izlaz: ne
Реч implementiratis је можда погрешно написана. Желите ли да реч "implementiratis" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč implementiratis: ['implementirati -> indeks: 1']
Unesite indeks: 1
Реч txt је можда погрешно написана. Желите ли да реч "txt" замените неком другом секвенцом карактера?
da|ne|izlaz: ne
Реч docx је можда погрешно написана. Желите ли да реч "docx" замените неком другом секвенцом карактера?
da|ne|izlaz: ne
Реч datoteksu је можда погрешно написана. Желите ли да реч "datoteksu" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč datoteksu: ['datoteku -> indeks: 1']
Unesite indeks: 1
Реч navedenii је можда погрешно написана. Желите ли да реч "navedenii" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč navedenii: ['navedeni -> indeks: 1', 'navedenim -> indeks: 2', 'navedenih -> indeks: 3']
Unesite indeks: 1
Реч eksportovanje је можда погрешно написана. Желите ли да реч "eksportovanje" замените неком другом секвенцом карактера?
da|ne|izlaz: ne
Реч snimitio је можда погрешно написана. Желите ли да реч "snimitio" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč snimitio: ['snimito -> indeks: 1', 'snimiti -> indeks: 2']
Unesite indeks: 2
Реч kjasan је можда погрешно написана. Желите ли да реч "kjasan" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč kjasan: ['klasan -> indeks: 1', 'kvasan -> indeks: 2', 'jasan -> indeks: 3', 'krasan -> indeks: 4', 'kasan -> indeks: 5']
Unesite indeks: 3
Реч implementiranje је можда погрешно написана. Желите ли да реч "implementiranje" замените неком другом секвенцом карактера?
da|ne|izlaz: ne
Реч tekstualu је можда погрешно написана. Желите ли да реч "tekstualu" замените неком другом секвенцом карактера?
da|ne|izlaz: da
Preporuke za reč tekstualu: ['tekstualnu -> indeks: 1']
Unesite indeks: 1
```

Слика 21: Интерактивни приказ седме команде

Покретањем осме команде се врши приказ првих неколико најучесталијих речи у тексту. Колико ће се речи приказати, зависи од уноса корисника. Такође, може се унети флег којим ће се мала и велика слова посматрати исто:

```
python diplomski_rad.py -f demo.docx -8 10
```

Слика 21: Приказ покретања осме команде

```
Prvih 10 najčešće korišćenih reči u tekstu:
1. i -> broj ponavljanja: 13
2. u -> broj ponavljanja: 8
3. alat -> broj ponavljanja: 6
4. se -> broj ponavljanja: 5
5. koje -> broj ponavljanja: 5
6. kao -> broj ponavljanja: 4
7. je -> broj ponavljanja: 4
8. pdf -> broj ponavljanja: 4
9. za -> broj ponavljanja: 3
10. datoteka -> broj ponavljanja: 3
```

Слика 22: Штампање првих 10 најучесталијих речи

```
python diplomski_rad.py -f demo.docx -8 10 -i
```

Слика 23: Приказ покретања осме команде уз спецификацију флега

```
Prvih 10 najčešće korišćenih reči u tekstu:
1. i -> broj ponavljanja: 13
2. u -> broj ponavljanja: 8
3. alat -> broj ponavljanja: 6
4. se -> broj ponavljanja: 5
5. koje -> broj ponavljanja: 5
6. računarski -> broj ponavljanja: 4
7. kao -> broj ponavljanja: 4
8. je -> broj ponavljanja: 4
9. nakon -> broj ponavljanja: 4
10. pdf -> broj ponavljanja: 4
```

Слика 24: Штампање првих 10 најучесталијих речи уз наведени флег

Тренутни изглед датотеке *demo.docx* је:

Diplomski rad

Tema: Računarski alat za jednostavnu obradu tekstualnih datoteka baziran na komandnoj liniji.

Zadatak:

Implementirati računarski alat koji se pokreće iz komandne linije i kao ulaz prima tekstualnu datoteku ekstenzije txt i *docx*. Neophodno je napraviti pomoćnu naredbu -p, *--pomoć* koja će korisniku dati jasan pregled svih mogućnosti koje navedeni alat nudi. Nakon izvršenih željenih izmena, izlazna datoteka se može snimiti pod ekstenzijom koju ima i ulazna (izmenjena) datoteka ili pak u *pdf* formatu.

Rezime rada:

Uvodni deo rada počinje sa upoznavanjem tehnologija koje su korišćene u radu i koje učestvuju u formiranju krajnje verzije računarskog alata. Pre dela koji je vezan za sam alat, njegovo implementiranje i korišćenje, čitalac će se takođe upoznati i sa pojmovima kao što su regularni izrazi, **Levenštajnovo rastojanje**. Nakon što se čitalac upozna sa tehnologijama i pojmovima koje je potrebno znati, sledi objašnjenje same logike svake stavke koju računarski alat nudi kao i detaljno objašnjenje vezano za njeno pokretanje.

Ključne reči:

Računarski alat, tekstualna datoteka, *Pajton*

FUNKCIONALNOSTI:

- Transformacija malih slova u velika gde god je to potrebno - početak rečenice, posle tačke.
- Nakon svakog zareza, ukoliko nema razmaka, dodati ga kao i nakon tačke.
- Ukloniti duple ili višestruke razmake između dve reči ili karaktera i reči.
- Transformacija iz latinice u ćirilicu i obrnuto.
- Eksportovanje datoteke u *pdf*.
- Mogućnost spajanja više pdf datoteka u jednu *pdf* datoteku.
- Najbliže pronađena reč.
- Reči koje se najčešće pojavljuju u tekstu.
- Pronađi prvo pojavljivanje unesene sekvence i zameni drugim unosom | Pronađi svako pojavljivanje unesene sekvence i zameni drugim unosom.

Слика 25: Уређена верзија рада

Последњу функционалност коју нуди софтверски алат јесте замена одређене секвенце знакова другом секвенцом знакова. У зависности од унетих флегова, извршиће се различити видови замена. На претходној слици је приказан уређен текст који садржи четири појављивања речи „Računarski“ – два пута је та реч написана великим и два пута малим почетним словом. Следи приказ четири могућа сценарија:

```
python diplomski_rad.py -f demo.docx -9 računarski kompjuterski
```

Слика 26: Приказ покретања команде девет без употребе флегова

Овако покренута команда ће заменити прву секвенцу *računarski* на коју наиђе новом секвенцом *kompjuterski*.

```
python diplomski_rad.py -f demo.docx -9 računarski kompjuterski -i
```

Слика 27: Приказ покретања команде девет са употребом једног флега

Резултат овакве команде ће бити замена прве пронађене секвенце *računarski* без обзира на разлику у малим и великим словима. У тексту са слике 25 ће се овом командом заменити секвенца са великим словом р – *Računarski*.

```
python diplomski_rad.py -f demo.docx -9 računarski kompjuterski -z
```

Слика 28: Приказ покретања команде девет са употребом једног флега

Овако покренута команда ће заменити сваку секвенцу *računarski* на коју наиђе новом секвенцом *kompjuterski*.

```
python diplomski_rad.py -f demo.docx -9 računarski kompjuterski -z -i
```

Слика 29: Приказ покретања команде девет са употребом оба флега

Резултат овакве команде ће бити замена сваке пронађене секвенце *računarski* без обзира на разлику у малим и великим словима. У тексту са слике 25 ће се овом командом заменити секвенце са великим словом р – *Računarski* и са малим словом р - *računarski*.

Закључак

Овим радом описан је софтверски алат базиран на командној линији, који пружа неколико функционалности у самој обради текстуалних датотека. Читалац је могао да се детаљно упозна са технологијама које су коришћене у оквиру овог рада, појмовима који су битни за потпуно разумевање дела рада који се односи на дескрипцију логике изворног кода. Такође, објашњен је сам поступак покретања наведеног софтверског алата.

Овакав вид софтверског алата може бити користан када је потребно брзо форматирање текста јер је потребно само навести улазну датотеку и функцију коју треба применити. Међутим, пошто данас доста људи који користе рачунар на дневном нивоу и користе алате за обраду текста нису баш информатички најпоткованији, следећи корак у развоју овог софтверског алата би био развој *GUI* (*Graphical User Interface*) апликације која ће имати јасно дефинисан интерфејс који пружа лако и једноставно коришћење.

Литература

[1]

www.python.org, *What is Python? Executive Summary*

[Приступљено у октобру 2019.]

Доступно на: https://www.python.org/doc/essays/blurb/

[2]

realpython.com, *How to Build Command Line Interfaces in Python With argparse*

Објављено: 12. јуна, 2019.

[Приступљено у октобру 2019.]

Доступно на: https://realpython.com/command-line-interfaces-python-argparse/

[3]

docs.python.org, *Regular expression operations*

[Приступљено у октобру 2019.]

Доступно на: https://docs.python.org/3/library/re.html

[4]

python-docx.readthedocs.io, *Python-docx*

[Приступљено у октобру 2019.]

Доступно на: https://python-docx.readthedocs.io/en/latest/

[5]

docs.python.org, *Access to underlying platform's identifying data*

[Приступљено у октобру 2019.]

Доступно на: https://docs.python.org/3/library/platform.html

[6]

numpy.org , *Numpy*

[Приступљено у октобру 2019.]

Доступно на: https://numpy.org/

[7]

pyi.org, *PySpellChecker*

[Приступљено у октобру 2019.]

Доступно на: https://pypi.org/project/pyspellchecker/

[8]

ghostscript.org, *What is Ghostscript?*

[Приступљено у октобру 2019.]

Доступно на: https://www.ghostscript.com/doc/9.27/WhatIsGS.htm

[9]

systutorials.com, *Paps Linux Man Page*

[Приступљено у октобру 2019.]

Доступно на: https://www.systutorials.com/docs/linux/man/1-paps/

[10]

libreoffice.org, *Edit all kinds of documents*

[Приступљено у октобру 2019.]

Доступно на: https://www.libreoffice.org/discover/writer/

[11]

github.com, *OfficeToPdf*

[Приступљено у октобру 2019.]

Доступно на: https://github.com/cognidox/OfficeToPDF

[12]

Jan Goyvaerts, *Regular Expressions*

Издато: у јуну 2007.

[Приступљено у октобру 2019.]

[13]

en.wikipedia.org, *Levensthein distance*

[Приступљено у октобру 2019.]

Доступно на: https://en.wikipedia.org/wiki/Levenshtein_distance

[14]

en.wikipedia.org, *Hamming distance*

[Приступљено у октобру 2019.]

Доступно на: https://en.wikipedia.org/wiki/Hamming_distance

[15]

mathworld.wolfram.org, *Triangle inequality*

[Приступљено у октобру 2019.]

Доступно на: http://mathworld.wolfram.com/TriangleInequality.html