Plan van aanpak - IPASS

Jimmy Bierenbroodspot

5 juni 2024

1 Probleembeschrijving

In de toekomst willen wij graag een AI-model hebben, dat de beste CV's op de markt kan maken voor onze klanten. Hiervoor hebben we natuurlijk veel data in de vorm van CV's nodig en gelukkig hebben we deze in grote aantallen. Nu is het probleem dat deze CV's in .pdf-formaat opgeslagen zijn en text hieruit halen is niet makkelijk [5].

We hebben het programma genaamd PyPDF[4] waarmee we in staat zijn om text uit .pdf-bestanden te lezen, het probleem is dat dit ongestructureerd is. Als voorbeeld kunnen we het voorbeeld CV in 1 nemen.

Figure 1: Voorbeeld van een Cliq CV

In 2 kunnen we zien wat er gebeurd als we de tekst uit 1 lezen. Dit ziet er uit als een onlogisch rommeltje. We kunnen met PyPDF echter ook de tekstopmaak kopiëren, zoals te zien in 3. Dit ziet er al beter uit maar als we bedenken dat dit allemaal op een regel staat, waarbij de nieuwe regels in dit geval juist worden getoond, maar eigenlijk aangeduid wordt met een \n.

```
Regio [WOONPLAATS]
DRIVE
[TEKST DRIVE - MIDDEN KOLOM
1 GEPOSITIONEERD]
KUNDE
MOGELIJKHEDEN
[TEKST MOGELIJKHEDEN - MIDDEN
KOLOM 1 GEPOSITIONEERD]OPLEIDING
Contactgegevens kandidaatERVARING
Student | startmaand startjaar - eindmaand eindjaarDRIE TREFWOORDEN +
PERSOONSOMSCHRIJVING Functietitel + bij + Bedrijf
[TEKST KUNDE - MIDDEN KOLOM
1 GEPOSITIONEERD]Titel opleiding + opleidinginstituut
[optionele uitleg studie][Meest effectieve zin van het hele CV, kort, krachtig, heel
erg passend bij de vacature.]
Functietiel + bij + Bedrijfstartmaand startjaar - eindmaand eindjaar
[Optionele uitleg van ervaring en ontwikkeling]
Student | startmaand startjaar - eindmaand eindjaarTitel opleiding + opleidinginstituut
[optionele uitleg studie]startmaand startjaar - eindmaand eindjaar
```

Figure 2: Voorbeeld van tekstextractie PyPDF

```
          DRIE TREFWOORDEN +
   PERSOONSOMSCHRIJVING
            Re   g   io    [WOONPL   AATS]


                                DRI   VE
   [TEKST DRIVE - MIDDEN KOL   OM
               1    GEPOSITIONEERD]
                               KUNDE
  [TEKST KUNDE - MIDDEN KOL   OM
               1    GEPOSITIONEERD]

              MOGEL   IJKHEDEN

[TEKST MOGEL   IJKHEDEN - MIDDEN
      KOL   OM 1    GEPOSITIONEERD]
```

```
ERVARING
F   u   n   ctietitel + b   ij + Bed   rijf
sta   rtma   a   n   d    sta   rtja   a   r - e   in   d   ma   a   n   d
[Me   e   st e   ffe   ctie   ve    zin    va   n    h   e   t h   e   le
e   rg    p   a   sse   n   d    b   ij d   e    va   ca   tu   re   .]
F   u   n   ctietiel + b   ij + Bed   rijf
sta   rtma   a   n   d    sta   rtja   a   r - e   in   d   ma   a   n   d
[Op   tio   n   e   le    u   itle   g    va   n    e   rva   rin   g    e
OPL   EIDING
T   itel o   p   leid   in   g    + o   p   leid   in   g   in   stitu   u
Stu   d   e   n   t | sta   rtma   a   n   d    sta   rtja   a   r - e   in
[o   p   tio   n   e   le    u   itle   g    stu   d   ie   ]
T   itel o   p   leid   in   g    + o   p   leid   in   g   in   stitu   u
Stu   d   e   n   t | sta   rtma   a   n   d    sta   rtja   a   r - e   in
[o   p   tio   n   e   le    u   itle   g    stu   d   ie   ]

Co   n   ta   ctg   e   g   e   ve   n   s ka   n   d   id   a   a   t
```

Figure 3: Voorbeeld van tekstextractie PyPDF met tekstopmaak

1.1 Waarom willen we dit?

In datawetenschap kennen we het concept GIGO, oftewel garbage in garbage out [2]. Als we aannemen dat ditzelfde zal gelden voor het model dat we gaan gebruiken dan zouden we graag willen dat onze invoerdata gestructureerd is.

Als we aannemen dat de tekst op een regel zou moeten komen te staan om ingelezen te worden door het model, dan zijn beide manieren van text extraheren met PyPDF niet gestructureerd. We willen dus dat de tekst op een logische volgorde komt te staan. Als een CV geen kolommen zou bevatten dan kunnen we met PyPDF de tekstopmaak kopiëren, nu weten we alleen van tevoren niet of een CV kolommen heeft of niet en als dat wel zo is weten we niet hoeveel kolommen er zijn. Om hierachter te komen moeten we naar het CV kijken en dat gaat niet met een geautomatiseerd proces.

2 Algoritme

In 3 kunnen we zien dat er een best duidelijke scheiding zit tussen de twee kolommen. Als we aannemen dat dit het geval is voor elke CV dan kunnen we een heuristiek opzetten om te bepalen hoeveel kolommen zich in een document bevinden. Vervolgens kunnen we deze informatie gebruiken om alle tekst op chronologische leesvolgorde onder elkaar te zetten.

2.1 Heuristiek

We kunnen de stappen voor de heuristiek als volgt op een rij zetten:

1. Haal de tekst uit het document d.m.v. PyPDF.
2. Vind de positie van de scheiding van de kolommen.
3. Plaats de kolommen onder elkaar met gebruik van de kolomscheiding.

Nu is het wat lastiger om erachter te komen waar de scheiding tussen de kolommen precies zitten. Hier komt het algoritme van pas.

2.2 Welk algoritme?

Wat we hier proberen te doen staat bekend als "page/document layout analysis", een techniek om de opmaak van een pagina te bepalen [3]. Omdat we in het huidige stadium alleen gebruik maken van bestaande LLM's, hoeven we geen algoritme te gebruiken om te begrijpen wat elk stukje tekst betekent. Het enige dat er voor nu echt toe doet is dat de tekst in een logische en chronologische volgorde staat. Ook willen we ons nog niet gaan focussen op CV's met een complexe opmaak.

Omdat we de tekstopmaak al in tekstvorm hebben (zie 3) lijkt het mogelijk om een relatief simpel clusteringsalgoritme, zoals Llyods algoritme [1], te gebruiken om samen met de elleboog methode om de kolommen in het document te vinden.

3 Taken

Voor dit project zijn de volgende taken benodigd:

3.1 Applicatie

1. Simpele UI d.m.v. Jupyter Notebook — Noodzakelijk
2. Een UI d.m.v. een front-end framework — Optioneel

3.2 Algoritme

1. Verkrijgen van tekst uit pdf-bestanden — Noodzakelijk
2. Implementeren van algoritme volgens bron — Noodzakelijk
3. Andere/betere algoritme implementeren — Optioneel

3.3 Rapportage

1. Probleembeschrijving — Noodzakelijk
2. Beschrijving van eisen — Noodzakelijk
3. Beschrijving van algoritme — Noodzakelijk
4. Evalueren van algoritme — Optioneel
5. Poster — Noodzakelijk

4 Planning

Week	**Taken**
23	- Implementeren van algoritme beginnen - Verkrijgen van tekst uit pdf-bestanden - Simpele UI d.m.v. Jupyter Notebook
24	- Implementeren afmaken - Zoeken naar andere algoritme - Evalueren van algoritme
25	- Betere UI - Rapport: Probleembeschrijving - Rapport: Beschrijving van eisen - Rapport: Beschrijving van algoritme
26	- Poster - Uitloop

Table 1: De planning

5 Risico's

5.1 Risico: afwijking van scope

Het is een risico dat er afgeweken kan worden van de originele scope en voorzie dat de kans dat dit gebeurt groot is. Dit betekent dat het mogelijk is dat er naar dingen gekeken gaan worden die niet leiden naar het verwachte resultaat.

De impact hiervan zou groot zijn aangezien dit kan leiden tot een incompleet product of missende onderdelen.

Dit kan vermeden worden door een duidelijk beeld te hebben van wat er gedaan moet worden en door en docenten op polshoogte te houden.

5.2 Risico: Incorrect ontwikkelen van het algoritme

Er is een risico dat het algoritme incorrect ontwikkeld kan worden en ik voorzie dat de kans hiervan gemiddeld is. De impact hiervan is groot.

Dit kan vermeden worden door een goed besef te hebben van de bron, pseudocode te gebruiken en tests voor de code te schrijven.

5.3 Risico: Inefficient algoritme

Er is een risico dat het algoritme Inefficient is, de kans dat dit gebeurt zal groot zijn. De impact hiervan is relatief klein aangezien er niet met al te veel data gewerkt hoeft te worden in de casus.

Dit kan vermeden worden door efficiënte datastructuren te gebruiken en zo min mogelijk code te schrijven.

References

[1] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (Mar. 1, 1982), pp. 129–137. DOI: 10.1109/tit.1982.1056489. URL: https://doi.org/10.1109/tit.1982.1056489.

[2] Merriam-Webster. *GIGO*. In: URL: https://www.merriam-webster.com/dictionary/GIGO (visited on 05/30/2024).

[3] L. O'Gorman. "The document spectrum for page layout analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.11 (Nov. 1993), pp. 1162–1173. ISSN: 1939-3539. DOI: 10.1109/34.244677.

[4] *pypdf*. Apr. 7, 2024. URL: https://pypi.org/project/pypdf/ (visited on 05/30/2024).

[5] Amit Timalsina. *How to extract data from PDF?* May 2, 2024. URL: https://www.docsumo.com/blog/extract-data-from-pdf (visited on 05/30/2024).

List of Figures

List of Tables