CHALMERS
UNIVERSITY OF TECHNOLOGY

Automatisk jobbmatchning

Matchning av kandidater och annonser med hjälp av Natural Language Processing

Examensarbete inom Data- och Informationsteknik

MARTIN KARLSSON
TOBIAS ROSENGREN

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Examensarbete
Göteborg, Sverige 2019

EXAMENSARBETE

Automatisk jobbmatchning

Matchning av kandidater och annonser
med hjälp av Natural Language Processing

Examensarbete inom Data- och Informationsteknik

MARTIN KARLSSON
TOBIAS ROSENGREN

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg, Sverige 2019

Automatisk jobbmatchning
Matchning av kandidater och annonser
med hjälp av Natural Language Processing
MARTIN KARLSSON
TOBIAS ROSENGREN

© MARTIN KARLSSON , TOBIAS ROSENGREN, 2019

Examensarbete
Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola / Göteborgs universitet
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain tex, pictures or other material that violates copyright law

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Chalmers Reproservice
Göteborg, Sverige 2019

Automatisk jobbmatchning
Matchning av kandidater och annonser
med hjälp av Natural Language Processing
Examensarbete inom Data- och Informationsteknik
MARTIN KARLSSON
TOBIAS ROSENGREN
Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola

SAMMANFATTNING

För att underlätta arbetsgången för rekryterare som dagligen arbetar med att hitta lämpliga uppdrag för sina konsulter skapades ett system som automatiskt kan matcha dessa konsulter mot olika uppdragsannonser. Att skapa ett sådant system var en komplex uppgift då uppdragsannonser ofta består av ostrukturerad text vilket är svårt att tolka för en dator. För att hantera detta problem tillämpades Natural Language Processing som är en metodik för att hantera ostrukturerad text med hjälp av artificiell intelligens och olika algoritmer. Det färdigställda systemet levererade flera positiva resultat, utöver dessa tillkom dock en del missvisande resultat. Systemets resultat rekommenderas att användas som riktlinjer att gå efter för en rekryterare när en lämplig kandidat ska väljas ut för ett uppdrag. Det finns mycket utrymme för utveckling för att förbättra systemets pålitlighet och minska sannolikheten för missvisande resultat.

Nyckelord: NLP, Datorlingvistik, Jobbmatchning

Automatic job matching
Matching of candidates and adverts using Natural Language Processing
Bachelor's thesis in Computer Science
MARTIN KARLSSON
TOBIAS ROSENGREN
Department of Computer Science and Engineering
Chalmers University of Technology

ABSTRACT

In order to facilitate the workflow of recruiters who work daily with finding suitable assignments for their consultants, a system was created that can automatically match these consultants against different job listings. Creating such a system was a complex task since job listings often consist of unstructured text which is difficult to interpret for a computer. To deal with this problem, Natural Language Processing was applied, which is a methodology for managing unstructured text using artificial intelligence and various algorithms. The completed system delivered several positive results, in addition to these some misleading ones were also produced. The system's results are recommended to be used as guidelines to pursue for a recruiter when a suitable candidate is to be selected for an assignment. There is much room for development to improve the system's reliability and to reduce the likelihood of misleading results.

Keywords: NLP, Computational Linguistics, Job Matching

INNEHÅLL

FÖRORD

Denna rapport är ett resultat av vårt examensarbete utfört under våren 2019 på Chalmers Tekniska Högskola. Vi rekommenderar att läsa den i sin helhet för att få en så klar bild som möjligt av dess innehåll.

Om man vill bli mer insatt i området datorlingvistik så rekommenderar vi att läsa kapitel 2: Teknisk Bakgrund. Detta ger även en bättre förståelse av många designval kring systemets utformning.

För att förstå hur systemet är konstruerat och vilka resultat som systemet producerat rekommenderar vi att läsa kapitel 4 till 6.

Om man främst är intresserad av vad systemets resultat innebär och hur systemet kan förbättras rekommenderar vi att läsa kapitel 6 och 8.

För den som ämnar att ge sig in på området datorlingvistik finns det otroligt många bra öppna resurser att ta del av. Flera av dessa citeras genom rapporten och har varit till mycket stor hjälp för att kunna genomföra detta arbete. Till alla de som tillägnat sin tid åt att göra dessa resurser tillgängliga för allmänheten vill vi ge ett stort tack.

Akronymer

AI Artificiell Intelligens
NER Named Entity Recognition
NLP Natural Language Processing
NLTK Natural Language Toolkit
POS Part Of Speech
Tf-idf Term frequency-inverse document frequency

Ordlista

Artificiell Intelligens Vetenskapen om intelligens i datorer
Artificiella Neurala Nätverk Samlingsnamn för en mängd självlärande algoritmer
Datorlingvistik Vetenskapligt område för behandling av språklig text och tal
Matchningsmetod En algoritm som utvärderar en likhet mellan en konsult och en uppdragsannons
PageRank Google's välkända rankningsalgoritm
Token En text delas i segment baserat på en logisk regel som resulterar i en lista med tokens
Värderingsmetod En algoritm som värderar ord

1 Inledning

Detta kapitel handlar främst om hur projektet tog sin form, hur mål och syfte togs fram samt vilka frågeställningar som uppkom kring projektets utformning. Vilka avgränsningar som gjordes tas även upp.

1.1 Bakgrund

Projektet utfördes åt ett konsultföretag inom IT och teknik som aktivt arbetar med att hitta nya uppdrag åt sina konsulter. Detta sker i dagsläget via en intern databas över deras egna konsulter där anställda manuellt matchar lämpliga konsulter mot olika typer av uppdrag. Dessa förmedlas normalt från olika typer av arbetsmäklare. Detta är ett mycket tidskrävande arbete i behov av effektivisering. För att uppnå en bättre lösning är tanken att konstruera ett nytt system som ska automatisera stora delar av denna process genom att ta fram lämpliga kandidater till uppdragen baserat på matchande kriterier hos konsulten i fråga och kraven specificerade i uppdraget.

1.2 Syfte

Genom att automatisera delar av rekryteringsprocessen så hoppas företaget kunna öka sin effektivitet och minska arbetsbelastningen hos de anställda som arbetar med detta i dagsläget. Med ett nytt system så skapas det även möjligheten att hantera fler annonser från fler mäklare, vilket kan ge konsulter inom företaget fler möjligheter när det kommer till att välja uppdrag. Ett nytt system skulle även kunna bidra till en kostnads- och arbetseffektivisering hos företaget då arbetstiden som krävs för matchningsprocessen kan komma att minska signifikant.

1.3 Mål

Projektets huvudsakliga mål är att skapa ett system som automatiskt kan matcha en anställd konsult från företagets databas mot ett utvalt uppdrag baserat på kraven och önskningarna i annonsen givet konsultens meriter. För att uppnå detta så måste systemet kunna hantera data från en databas med konsultprofiler samt jämföra och matcha denna mot data som representerar en uppdragsannons.

1.4 Frågeställningar

För att kunna nå projektets mål finns det många frågor som kräver svar för att kunna utforma en skalbar och tillfredsställande lösning.

- Hur kvantifieras systemets effektivitet?
- Vilka programspråk och bibliotek lämpar sig bäst för att utforma applikationen?
- Hur ska data från databasen och annonser förbehandlas?
- På vilket sätt ska matchningsprocessen ske?
- Vad räknas som en bra matchning?
- Vilka metoder finns för att bearbeta löpande text som annonser ofta består av?
- Bör artificiell intelligens tillämpas?
- Vad begränsar systemets skalbarhet?

1.5 Avgränsningar

Det kommer inte ligga fokus på att skapa någon form utav grafiskt användargränssnitt eller kommandotolksverktyg för programvaran. Vad gäller metoder för själva matchningsprocessen kommer färdiga programbibliotek att brukas för sådant som exempelvis artificiella neurala nätverk, webbskrapning och datorlingvistik. All programvara och relaterad dokumentation kommer att skrivas och dokumenteras på engelska. På grund av tidsramen för projektet så kommer endast konsultprofiler och uppdragsannonser på engelska att bearbetas då bearbetning av dokument på flera språk kan öka komplexiteten.

2 Teknisk bakgrund

I detta kapitel presenteras de idéer och koncept som nyttjats för att implementera delar av systemets komponenter, vilka algoritmer som använts samt vilka olika underliggande teknologier som ligger till grund för våra designval som diskuteras ytterligare i metodkapitlet. Detta för att skapa en ökad förståelse för resonemangen kring projektets utformning. Kapitlet kommer även att gå igenom hur indatan i form av konsultprofiler och uppdragsannonser är formaterade.

2.1 Indata

Konsultprofiler kommer primärt att hämtas från excel-dokument där en rad representerar en konsult med ett id, kärnfärdigheter, tidigare arbetsområden och kunskaper. För att förklara vad detta betyder ges några exempel och förklaringar:

Id - Ett unikt nummer för varje konsult.
Kärnfärdigheter - Project Management, Software design, Data analytics.
Tidigare arbetsområden - Manufacturing, Design, Development, Research.
Kunskaper - C++, Java, Test Automation, ITIL, SAP, Autosar, CAD, Scrum Master.

Uppdragsannonser kommer att se ut likt följande exempel med mer eller mindre detaljer:

Software Developer

For our client we are looking for a Software Developer.

Job description:

We are looking for a rockstar developer who works best under pressure. The work will be carried out day and night to reach deadlines and satisfy our customers. The office you will be working from has multipe beanbags and a great view over our breathtaking parking lot filled with self driving cars and scooters. Knowledge of databases is expected.

You will be part of an agile and self-organized team of developers, architects and testers that is responsible for all elements of application and service development. You will be involved in requirements work, architecture, development, testing, release management and operation.

Required skills:

- *Java or JavaScript*
- *Lego*
- *Scratch*

2.2 Artificiella neurala nätverk

En stor del av projektet handlar om hur man kan behandla löpande text med en dator. Detta då en jobbannons eller en konsultprofil brukar vara utformad som en löpande text helt och hållet eller med vissa inslag av det. I vilket fall som helst finns det ingen vedertagen standard för hur en jobbannons eller en konsultprofil kan se ut. Detta gör det svårtolkat för en dator som är mycket bättre lämpad att hantera strukturerad data med en tydlig ordning eller hierarkisk struktur. För att hantera ostrukturerad text har det skapats olika lösningar med hjälp av artificiella neurala nätverk [1]. Ett eget artificiellt neuralt nätverk tränades även för att hitta färdigheter i text, detta beskrivs mer i detalj i kapitel 2.6.

2.3 Natural language processing

NLP är ett subfält till AI och datorvetenskap och behandlar bland annat området kring hur en dator kan tolka information som förmedlas genom mänskligt skriven text [2]. Som tidigare nämnts har det använts ett redan tränat nätverk för att hantera vissa problem som behövde lösas för att kunna behandla dokument i form av konsultprofiler och jobbannonser. Det nätverket som använts är en del av ett open-source programbibliotek skrivet i Python [3] vid namnet NLTK (Natural Language Toolkit) [4].

2.4 Natural Language Toolkit

De mest intressanta egenskaperna hos NLTK för detta projekt var möjligheten att kunna förbehandla dokument och på så vis göra dem enklare att arbeta med för en dator. Den funktionalitet som användes beskrivs ytterligare i dessa underkapitel.

2.4.1 Part of speech tagging

POS-tagging görs för att gå igenom ett dokument och markera varje ord med sin respektive POS-tagg. Detta görs för att kunna dela upp de i ordklasser som till exempel verb eller substantiv. Detta demonstreras bäst med ett exempel:

```
text = "There is a lady who is sure all that glitters is gold"
tokens = nltk.word_tokenize(text)
nltk.pos_tag(tokens)

output:
[('There', 'EX'), ('is', 'VBZ'), ('a', 'DT'), ('lady', 'NN'), ('who', 'WP'),
('is', 'VBZ'), ('sure', 'JJ'), ('all', 'PDT'), ('that', 'DT'),
('glitters', 'NNS'), ('is', 'VBZ'), ('gold', 'JJ')]
```

Varje ord har nu tilldelats en POS-tagg som beskriver vilken typ av ord det är. ‘Lady’ har till exempel fått taggen ‘NN’, vilket innebär att det är ett substantiv. Detta kan sedan användas för att utföra ytterligare operationer på dokumentet som kräver att varje ord har en POS-tagg.

2.4.2 Lemmatisering

Att lemmatisera ett ord innebär att reducera det till sin basform eller sitt så kallade lemma. Exempelvis så skulle orden “Playing, Plays, Played” efter lemmatisering alla reduceras till sitt lemma “Play”. För att kunna utföra denna operation krävs dock att varje ord har associerats med en POS-tagg. Anledning till detta är för att enkelt kunna gruppera samman ord som alla har samma basform och därmed oftast samma innebörd.

2.4.3 Stopwords

Stopwords är ord som inte på något direkt sätt bidrar till en texts huvudsakliga budskap. Exempel på stopwords skulle vara “och, där, så, som, till”. I NLTK finns det en lista med stopwords som kan användas för att filtrera ut dessa ur en text. Detta koncept kan också demonstreras mycket bra med ett tydligt exempel på en text före och efter att stopwords har eliminerats:

```
text = "There is a lady who is sure all that glitters is gold"
remove_stop_words(text)

Output: ['there', 'lady', 'sure', 'glitters', 'gold']
```

Texten har fortfarande bibehållet mestadelen av sitt budskap och samtidigt har onödiga ord eliminerats för att ytterligare underlätta processen med att plocka ut värdefulla nyckelord ur texten.

2.5 Värdering av nyckelord

I texter finns det ord som har mer eller mindre betydelse för textens budskap. De ord som har större betydelse för textens budskap kommer i denna rapport att beskrivas som nyckelord. Nyckelorden i sig kan även att behöva rangordnas enligt de mest betydelsefulla för att få ut endast de viktigaste av nyckelorden. Att rangordna ord kräver att vi ger ett sorts värde och sorterar orden enligt dess värdering. Värderingen av ett ord kan även uttryckas som en vikt eller en rank.

Om vi som människor går igenom en lista med nyckelord så kan vi spontant plocka ur de mest värdefulla med hjälp av vår förståelse av språket vi läser. När detta ska uträttas av en dator är det inte självklart hur man ska avgöra detta värde. Det finns ett flertal erkända metoder för att lösa detta problem med varierande resultat och olika anpassningar baserat på vilken typ av text det rör sig om. I detta kapitel tas de metoder upp som implementerades och testades inom projektets ramar. Det finns naturligtvis ännu fler metoder än de som nämns här. De som tas upp för detta projekt är några av de vanligast förekommande.

2.5.1 Frekvensbaserad värdering

Med en frekvensbaserad metod för värdering av nyckelord ges varje ord en vikt som representeras av hur många gånger ordet förekommer i en text. Det vill säga att om ett ord förekommer ofta ges det hög vikt och anses därmed vara ett viktigt nyckelord. Fördelen med denna metod är att den är mycket enkel att implementera och att många metoder för att värdera nyckelord använder sig av någon form av frekvensbaserad tilldelning av vikter i delar av sina algoritmer. Nackdelar med en rent frekvensbaserad metod är dock att den i sig ofta producerar näst intill meningslösa resultat då vanligt förekommande ord i en text oftast inte är de som ger texten sitt budskap.

2.5.2 Tf-idf

En utveckling av frekvensbaserad värdering är *tf-idf* algoritmen, kort för *Term frequency-inverse document frequency* [5]. Algoritmen används för att beskriva hur viktigt ett ord är i ett dokument i en samling av dokument genom att tilldela ord vikter. En större vikt betyder att ett ord är viktigare i dokumentet än ett annat ord med en mindre vikt.

För att räkna ut hur stor tf-idf vikten är för ett ord måste man först räkna ut en *Term frequency* (tf) och en *Inverse document frequency* (idf) som sedan multipliceras enligt:

$$tfidf = tf * idf$$

Term frequency är hur många gånger ett ord förekommer i ett dokument dividerat med antalet ord i dokumentet. Det vill säga att ett ord som förekommer många gånger i ett dokument får ett högt värde.

Inverse document frequency räknas ut genom att ta det logaritmiska värdet av divisionen av antalet dokument och antalet dokument som ordet existerar i. Detta ger därmed ett högt värde till ovanliga ord och ett lågt värde till ofta förekommande ord.

Tf-idf algoritmen kan även beskrivas matematiskt enligt följande formel:

$$w_{a,b} = \frac{n_{a,b}}{c_b} * log(\frac{N}{d_a})$$

$w_{a,b}$ = tf-idf vikten för ord a i dokument b
$n_{a,b}$ = förekomster av ord a i dokument b
c_b = antal ord i dokument b
d_a = förekomster av ord a i alla dokument
N = antal dokument

Sammanfattningsvis så kan tf-idf beskrivas som en algoritm som belönar ord med en större vikt om de är ovanliga eller om de är ofta förekommande i ett dokument. Resultatet av algoritmen påverkas även väldigt mycket beroende på hur många och vilken typ av dokument som behandlas då till exempel ovanliga ord inte blir så ovanliga om endast liknande dokument inom samma genre hanteras.

2.5.3 TextRank

TextRank är en oövervakad och grafbaserad algoritm som kan användas för att värdera till exempel meningar och nyckelord i texter [6]. Algoritmen avgör värdet av ord eller meningar i texter genom att behandla dessa som noder i en graf och med hjälp av kanter mellan dessa kan Google's välkända rankningsalgoritm PageRank användas för att räkna ut en rank för varje nod i grafen. Hur dessa kanter skapas mellan noder i grafen är upp till implementationen av TextRank och har stor betydelse för hur resultatet av algoritmen kommer att se ut. Dessa kanter kan skapas mellan till exempel närliggande ord, ord i samma genre, språkligt liknande ord eller mellan ord med andra relationer. Oavsett hur noder definieras och hur kanter skapas i grafen så följer TextRank följande steg:

1. Identifiera vilken enhet som bäst representerar en nod för uppgiften i fråga och lägg till de som noder i grafen. Enheten kan till exempel vara ord, meningar, paragrafer eller liknande. Väljs enheten meningar så får varje mening en vikt och på samma sätt får varje ord en vikt om ord väljs som enhet.

2. Identifiera relationer mellan noderna och skapa kanter. Dessa kanter kan vara riktade, oriktade, viktade eller oviktade.

3. Iterera en rankningsalgoritm tills konvergens uppnås. Detta är vanligtvis PageRank.

4. Sortera noderna baserat på rank givet av rankningsalgoritmen.

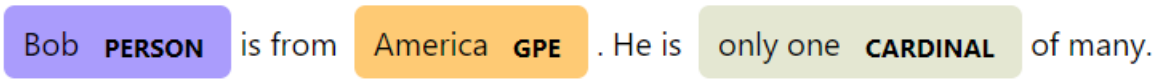
Sammanfattningsvis så är TextRank en flexibel algoritm som inte kräver stora mängder data för att leverera användbara resultat.

2.6 Named Entity Recognition

För att hitta nyckelord i texter utan att ge orden vikter testades en metod känd som *Named Entity Recognition*, även känt som NER [7]. Metoden går ut på att använda ett artificiellt neuralt nätverk som är tränat för att hitta ord och stycken i texter som passar in i specifika kategorier beroende på sammanhanget. Det finns redan tränade nätverk öppet tillgängliga som kan hitta ord och stycken för kategorier som namn, länder, adresser, organisationer med mera. Det finns dock inget öppet tillgänglig nätverk för att hitta färdigheter, vilket kan vara användbart för att hitta nyckelord i annonser och dylikt.

För att träna ett nätverk så att det kan hitta ord och stycken som passar i nya kategorier så behövs träningsdata som är markerad med vad för kategori som ett stycke tillhör. Att få tag i stora mängder av bra träningsdata är ofta komplicerat och kan helt avgöra hur bra nätverket kommer att prestera. Dessa tränade nätverk kan även beskrivas som *modeller*.

Denna bild presenterar hur ett av spaCy's tränade nätverk har markerat delar av en text [8].



PERSON - Namn på personer
GPE - Geopolitical entity, dvs. länder, städer, orter
CARDINAL - En typ av antal

2.7 Webscraping

Konceptet webscraping handlar om att snabbt plocka mycket information från en hemsida med hjälp av mjukvara, ofta i form av ett skript [9]. Webscraping användes för att slippa manuellt leta igenom och hämta ner flera hundra jobbannonser, vilket hade varit mycket tidskrävande och ineffektivt. Hur en webscraper designades och implementerades beskrivs i kapitel 4.2.

2.8 Matchning med hjälp av nyckelord

När nyckelord har extraherats från en uppdragsannons och en konsultprofil ska dessa matchas på ett sätt som ger en indikation på hur väl lämpad konsulten i fråga är för det givna uppdraget baserat på konsultens färdigheter gentemot kraven i annonsen. I detta avsnitt beskrivs de olika metoder som implementerats och testats för att skapa en matchning samt deras för- och nackdelar. Implementationen av varje metod beskrivs mer ingående i kapitel 4.4 Matchningsverktyget.

2.8.1 Snittmängd

Med dessa metoder avgörs hur bra en matchning är genom att helt enkelt titta på förhållandet mellan storleken på olika mängder. Den ena mängden är nyckelorden som tagits från konsultprofilen och den andra mängden är nyckelorden från uppdragsannonsen. Resultatet av matchningen blir i detta fall en procentsats som ges av hur stor den ena mängden är i förhållande till den andra.

2.8.2 Cosinus-likhet

Denna metod avgör hur bra en matchning är genom att bilda två vektorer givet två olika dokument. I detta projektets fall är det första dokumentet en uppdragsannons och det andra en konsultprofil. För att kunna forma vektorerna krävs att dokumentet innehåller information om varje nyckelord och dess vikt, vikten har räknats ut med hjälp av antingen TextRank [6] eller tf-idf [5]. Detta kan enkelt demonstreras med ett exempel på två dokument som ska jämföras med denna metod:

```
Dokument 1:          Dokument 2:
once    0.25         once    0.25
upon    0.25         upon    0.25
a       0.1          the     0.7
time    0.3          time    0.2
```

Baserat på dessa två dokument bildas vektorerna nedan:

Ord:	once	upon	a	the	time
Vektor A:	0.25	0.25	0.1	0	0.3
Vektor B:	0.25	0.25	0	0.7	0.2

Där ett ord endast förekommer i det ena av dokumenten ges det värdet 0 i vektorn som representeras av det andra dokumentet, på så vis som “a” och “the” har fått värdet 0 i sina vardera vektorer.
För att sedan avgöra hur väl dokumenten matchar varandra räknas detta ut med formeln:

$$\frac{A \cdot B}{||A|| \cdot ||B||} = 0.4819$$

Resultatet ligger i intervallet 0 till 1 för vektorer med enbart positiva element och representerar hur lika eller olika dokumenten är [10]. Ett högre värde tyder på är de är mer lika och en mindre värde tyder på att de är mindre lika.

3 Metod

I detta kapitel beskrivs arbetsgången och hur olika komponenter utformades och testades. Det behandlar även hur viss data insamlades och hur denna kunde behandlas på olika sätt med varierande resultat.

3.1 Arbetsgång

Arbetet skedde med en scrumbaserad arbetsmetodik [11]. Hela scrum-systemet nyttjades dock ej eftersom utvecklingsgruppen bestod av två personer vilket medförde svårigheter att utnyttja scrum fullt ut. Verktyget Trello användas för att representera gruppens product backlog [12]. Arbetet skedde huvudsakligen i grupp på gemensam plats för att underlätta samarbete och diskussion. Med platformen GitHub [13] och med hjälp av arbetsflödet GitFlow [14] möjliggjordes parallell utveckling vilket ökade produktiviteten i gruppen.

3.2 Utformning och testning

För att kunna utforma systemet behövde ett flertal komponenter designas och implementeras med ett bra programflöde och med utrymme för skalbarhet.

De huvudsakliga egenskaperna som krävdes av komponenterna i applikationen var:

- Förbehandling av konsultprofiler och uppdragsannonser
- Rankning av nyckelord
- Hämtning av uppdragsannonser från mäklare
- Matchning av konsultprofiler och uppdragsannonser
- Representation och presentation av resultat

För att komma fram till så optimala lösningar som möjligt för varje komponent så testades dessa på olika sätt. Bland annat genom att ändra i vilken sekvens olika operationer utfördes under förbehandlingen av dokumenten.

För att testa de olika rankningsalgoritmerna användes de på varierande mängder av dokument och olika typer av dokument såsom annonser riktade mot programmerare blandat med annonser riktade mot andra typer av uppdrag, till exempel logistik eller kemi.

Hämtning av data i form av uppdragsannonser krävde ingen direkt testning. Om datan kunde hämtas uppfyllde komponenten sitt syfte och annars inte.

Testning av matchningsalgoritmerna skedde genom att kombinera olika rankningsalgoritmer på olika typer av dokument för att sedan jämföra resultaten.

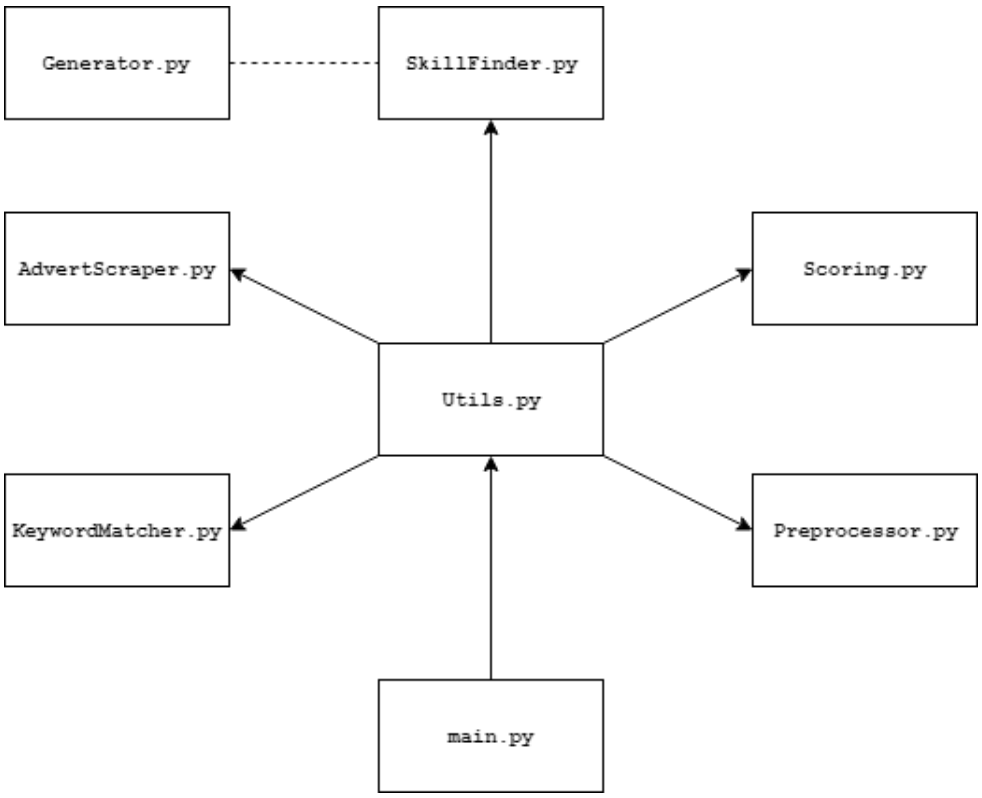
Presentation av resultat skedde huvudsakligen genom att skriva resultaten av en matchningsalgoritm till olika filer med vilka konsulter som matchades med vilka resultat mot olika annonser.

4 Systemkonstruktion

I detta kapitel beskrivs hur systemet byggdes upp från grunden. Bland annat beskrivs hur projektets bas är strukturerad och hur de olika komponenterna utformades och utvecklades under projektets gång.

Systemet är konstruerat enligt schemat i figur 4.1. Utils är implementerad så att den kombinerar metoder från de andra modulerna för att utföra större operationer såsom fullständig matchning av samtliga konsulter mot samtliga annonser. Dessa metoder har sedan använts i main för att skapa alla resultatfiler.

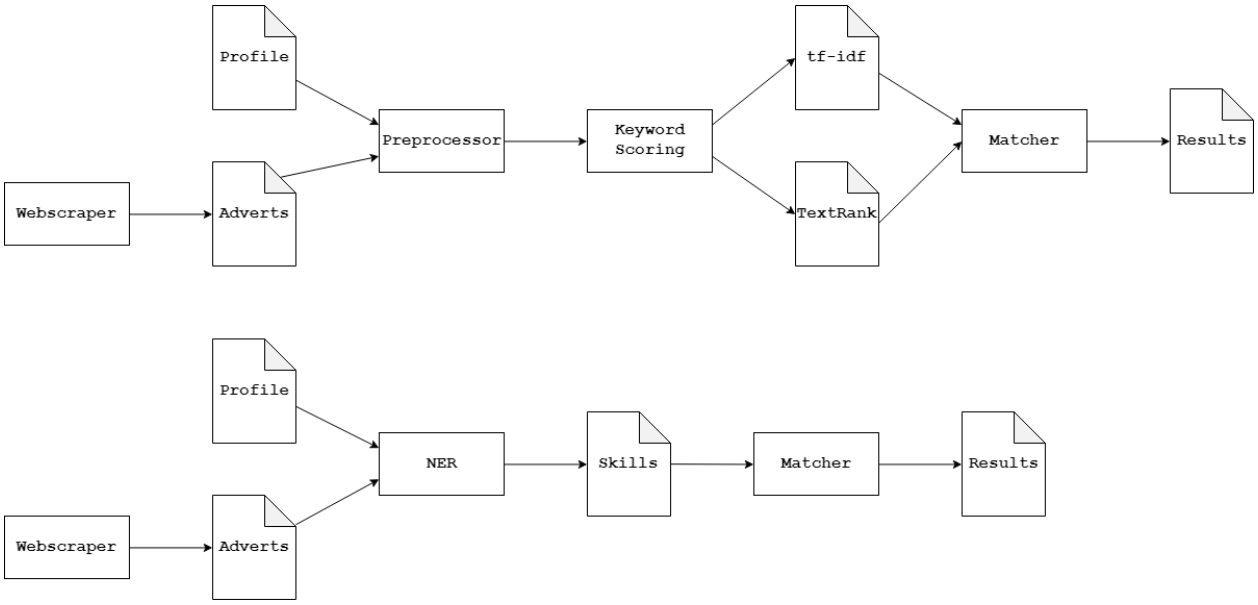
Varje komponent beskrivs utförligt i sitt eget avsnitt. Värt att notera är att Generator-modulen är helt fristående och endast skapar data som SkillFinder i sin tur använder sig av.



Figur 4.1: Systemöversikt.

Programflödet sker enligt följande. Först hämtas uppdragsannonser via en webscraper. Sedan körs annonserna och konsultprofilerna genom preprocessorn eller ett NER-nätverk. De dokument som går genom preprocessorn får sedan alla sina nyckelord värderade med tf-idf och TextRank. De dokument som går genom NER-nätverket får sina relevanta färdigheter extraherade till nya dokument.

Alla dessa körs sedan genom matchningsverktyget med tre olika matchningsmetoder vilket ger ett slutgiltigt resultat på sammanlagt 8 filer. Dessa innehåller varje annons id samt de bästa konsulterna som matchades mot varje annons och en procentsats som representerar hur bra matchningen är baserat på vilken matchningsalgoritm som användes. Flödet återfinns i figur 4.2.



Figur 4.2: Flödesschema av systemet. Den övre delen använder vikter och den undre använder *Named Entity Recognition*.

Ett exempel på hur en post från en resultatfil kan se ut. Överst är annonsens id, därefter följer de bäst matchande konsulternas id samt hur bra matchningen är:

```
--------------------------------
Advert id: 1

Consultant: 1, Match: 59.0%
Consultant: 2, Match: 14.0%
Consultant: 3, Match:  7.0%
Consultant: 4, Match:  5.0%
Consultant: 5, Match:  5.0%
Consultant: 6, Match:  5.0%
--------------------------------
```

4.1 Projektbas

Kodbasen är skriven i programspråket Python, detta då Python lämpar sig mycket väl för datahantering och har en hög abstraktionsnivå vilket har underlättat stora delar av arbetet.

4.2 AdvertScraper.py

För att kunna hämta uppdragsannonser på ett effektivt sätt konstruerades en webscraper. Denna implementeras mot en enda hemsida och är därför inte speciellt generell. Den uppfyllde dock sitt syfte med att hämta data i form av uppdragsannonser som sedan kunde användas för att testa olika matchnings- och rankningsalgoritmer på.

4.2.1 Implementation

Webscrapern designades som en Python-klass med flera olika metoder för att hämta data från en hemsida. Stora delar av klassen använder sig av Python-biblioteket BeautifulSoup [15] som är ett verktyg för att underlätta skrapning av hemsidor genom att ta bort html-taggar och dylikt. Anledningen till att detta bibliotek användes var för att det är ett mycket populärt bibliotek som består av färdiga metoder för vanliga problem relaterade till webscraping vilket sparade mycket tid.

För att kunna hämta vissa delar av hemsidor som laddas in dynamiskt med till exempel JavaScript [16] användes verktyget Selenium [17] som är ett verktyg för att automatiskt styra en webbläsare, vilket möjliggjorde hämtning av dynamiska element.

Slutligen implementerades en metod för att kunna hämta samtliga annonser belägna på en hemsida på en gång. Detta gjorde det mycket enkelt att samla in bra data snabbt och effektivt, för att kunna använda denna för att testa andra komponenter på.

4.3 Preprocessor.py

Syftet med denna komponent var att förbehandla dokument i form av konsultprofiler och uppdragsannonser. Som tidigare nämnt i teknisk bakgrund används POS-taggning, lemmatisering och eliminering av stopwords för att förbehandla text för att få ut en lista med icke-rankade nyckelord som kan användas av övriga komponenter.

4.3.1 Implementation

Preprocessorn är designad så att den arbetar stegvis med att förbehandla en sträng. Tanken är att denna sträng representerar ett dokuments innehåll och förbehandlingen sker enligt följande sekvens:

1. Strängen delas upp i tokens.
2. Varje token som är ett stopword eller en oönskad symbol tas bort.
3. Varje token POS-taggas för att sedan lemmatiseras.

Till en början så användes NLTK för att dela upp strängen i tokens men efter testning framgick det att önskvärda nyckelord såsom "C#" och ".NET" modifierades till "C" och "NET" vilket ger orden en annan betydelse. På grund av detta så utvecklades en egen funktion för att dela upp strängen i token med hjälp av reguljära uttryck även känt som regex [18]. Den egna funktionen börjar med att använda regex för att subtrahera bort tecken i strängen som är oönskade och sedan delas strängen upp vid blanksteg och returnerar en lista med tokens.

För att bestämma vad som klassas som stopwords användes delvis NLTK för att bestämma typiska engelska stopwords och även en egen lista skapades med diverse oönskade ord för att komplementera NLTK. Denna lista ger flexibilitet för att enkelt lägga till ord under utvecklingen och den framtida användningen som sedan filtrerar bort token på samma sätt som stopwords filtreras ut. En lista med symboler användes även för att eliminera oönskade tokens.

Efter att listan av tokens har filtrerats från stopwords och symboler så POS-taggas och lemmatiseras varje token. För POS-taggning användes ett tränat nätverk från NLTK och för lemmatisering användes WordNetLemmatizer, även denna från NLTK.

Experiment utfördes för att avgöra i vilken ordning steg 2 eller 3 skulle ske då de levererade samma resultat. Utifrån en tidsanalys mellan ordningarna så bestämdes det att stopwords och oönskade symboler skulle elimineras innan lemmatisering för att undvika onödig beräkningstid av att lemmatisera ord som ändå skulle elimineras i nästa steg. Detta har dock endast betydelse vid stora mängder stopwords och oönskade symboler.

4.4 Scoring.py

För att kunna bestämma de viktigaste nyckelorden i den givna texten krävdes det att rankningsalgoritmer implementerades för att kunna rangordna nyckelorden. Detta gjordes i en egen komponent för att möjliggöra skalbarhet och flexibilitet för framtida utveckling. Det skapar även möjligheten att återanvända komponenten i andra sammanhang då den har låg koppling och hög kohesion.

4.4.1 Implementation

Utvecklingen av komponenten började med att implementera tf-idf algoritmen som beskrivs tydligare i teknisk bakgrund. Tf-idf implementerades med tre olika funktioner för att skapa flexibilitet och för att göra det möjligt att använda endast delar av den för testning och i andra fall då funktionerna kan vara användbara. Den första funktionen räknar ut *Term frequency*, den andra räknar ut en *Inverse document frequency* och den tredje binder dessa samman och räknar därmed ut *Term frequency-inverse document frequency* vikten för varje nyckelord i varje givet dokument.

Den andra algoritmen som implementerades var TextRank som även den beskrivs tydligare i teknisk bakgrund. Implementeringen av TextRank använder sig av biblioteket NetworkX för att skapa en graf av givna nyckelord och för att utföra PageRank på denna graf [19]. I denna implementation av TextRank skapas bågar mellan alla direkt närliggande ord.

Värt att nämna är att i det engelska språket så används så kallade “multiword expressions” där flera ord med blanksteg mellan varandra kan användas som ett ord [20]. Ett exempel på ett engelskt multiword är “traffic light” där det menar trafikljus men även skulle kunna tolkas som trafik och ljus separat om orden delats upp. Detta gör valen av nyckelord mycket mer komplicerade då multiword måste identifieras och tas i åtanke. För att förenkla detta så har de tidigare nämnda implementationerna valt att hantera alla dessa multiword som separata ord.

4.5 KeywordMatcher.py

För att kunna matcha en konsult mot ett uppdrag krävdes en komponent med egenskapen att kunna räkna ut hur pass bra en matchning mellan en konsult och ett uppdrag är. Sättet en rekryterare hanterar denna process är mycket annorlunda mot hur denna komponent gör det. En rekryterare kan jämföra och analysera alla delar av kraven i en annons och förstå samband såsom koppling mellan erfarenhet och olika färdigheter hos en konsult för att fatta ett beslut om en konsult är lämplig för ett givet uppdrag. Detta kan inte denna komponent göra.

Komponenten är istället uppbyggd så att den använder olika algoritmer för att avgöra hur bra en matchning är. Dessa algoritmer är uppbyggda på olika sätt och lägger vikt på olika saker för att avgöra hur pass bra en matchning är. Varje algoritm tar en konsultprofil och en uppdragsannons som input, dessa har gått genom preprocessorn först. Resultatet av varje algoritm beskrivs rent konkret i detta avsnitt, hur representativt resultatet är för en bra matchning diskuteras i kapitel 7. Algoritmerna presenteras som pseudokod och med en beskrivning i ord.

4.5.1 SubsetMatch

Algoritmen bildar två mängder, den ena innehåller alla ord ur konsultprofilen (mängd k), den andra innehåller alla ord från uppdragsannonsen (mängd u). Därefter räknas det ur hur många av orden i u som återfinns i k. Algoritmen sker enligt följande flöde:

```
1  subsetMatch(file k, file u):
2      k_set = createSet(k.getKeys)
3      u_set = createSet(u.getKeys)
4
5      count = 0
6      for key in u_set:
7          if key in k_set:
8              count++
9
10     return count / length(u_set)
```

Resultat är en procentsats av hur många nycklar i mängden u som återfinns i mängden k dividerat med antalet element i mängden u. Detta är den enklaste algoritmen bland de som utvecklades. Denna algoritm kräver ej viktade nyckelord.

4.5.2 JaccardMatch

Denna algoritm bildar två mängder på samma sätt som SubsetMatch. Därefter tittar den på snittet mellan de två mängderna och returnerar resultatet som en procentsats av hur stort snittet är jämfört med storleken på unionen av mängderna, denna metod är känd som Jaccard index [21]. Algoritmen sker enligt följande flöde:

```
jaccardMatch(file k, file u):
    k_set = createSet(k.getKeys)
    u_set = createSet(u.getKeys)

    union_set = k_set.union(u_set)
    intersect_set = k_set.intersection(u_set)

    return length(intersect_set) / length(union_set)
```

Även detta är en relativt enkel algoritm och kräver ej viktade nyckelord.

4.5.3 CosineMatch

Denna algoritm bildar två Python-baserade datastrukturer som kallas dictionaries. Ett dictionary är en datastruktur som består av nyckel-värde par. Nycklarna i dessa dictionaries formas av nyckelorden i dokumenten som tas som inparametrar och värden tas från vikten som är associerade med varje ord och har räknats ut med hjälp av rankningsverktyget. Ett sådant dokument kan se ut som följande för att tydliggöra med ett exempel:

Nyckel	Värde
cloud	0.034021823201129 01
start	0.02341294424680349
security	0.019392417300670983
innovation	0.019047501231721872
software	0.016579698578909847
environment	0.015903267500331855

Denna algoritm är något mer komplex än de tidigare nämnda och returnerar ett värde som representerar hur pass lika de bildade vektorerna är ur ett geometriskt perspektiv. Algoritmen arbetar enligt följande flöde och använder sig av cosinus-likhet [10]:

```
cosineMatch(file k, file u):
    k_dict = buildDictionary(k)
    u_dict = buildDictionary(u)

    union_set = createSet(k_dict.keys).union(createSet(u_dict.keys))

    k_vector = []

    for word in union_set:
        if word in k_dict:
            k_vector.append(k_dict[word])
        else:
            k_vector.append(0)

    # Form the vector u_vector in the same way as k_vector

    return dot_product(k_vector, u_vector) /
           (normalize(k_vector) * normalize(u_vector))
```

Resultatet av algoritmen blir alltså skalärprodukten mellan vektorn k och vektorn u dividerat med produkten av de normaliserade vektorerna. Två vektorer som har samma komponenter kommer att producera resultatet 1 medans två vektorer som inte delar några komponenter kommer få resultatet 0, detta gäller för vektorer där samtliga element är större än 0.

4.6 Generator.py

Denna modul är designad för att generera träningsdata för NER-modellen i SkillFinder. Först delas alla annonser upp i meningar. Annonserna har hämtats med hjälp utav webskraparen. Sedan gås varje mening igenom och varje ord jämförs mot en lista med vanliga färdigheter, om ett ord matchas mot listan markeras dess start- och slutindex ut tillsammans med en markering för vad för sorts entitet det är, i detta fall märks de med markeringen ”SKILL”. Ett exempel på träningsdata som genereras baserat på meningen nedan:

”For our client we are looking for an Application Software Architect Job description: We are looking for an Application Software Architect to work with Cloud environments within automotive.”

```
TRAIN_DATA = [
    (
    """For our client we are looking for an Application Software Architect
    Job description: We are looking for an Application Software Architect
    to work with Cloud environments within automotive.""",
    {"entities": [(8, 14, 'SKILL'), (37, 48, 'SKILL'), (49, 57, 'SKILL'),
    (58, 67, 'SKILL'), (107, 118, 'SKILL'), (119, 127, 'SKILL'),
    (128, 137, 'SKILL'), (151, 156, 'SKILL'), (177, 187, 'SKILL')]},
    ),
]
```

4.7 SkillFinder.py

Denna modul användes för att träna och använda en modell som kan hitta tekniska färdigheter i löpande text. Detta användes som ett alternativ till de tidigare nämnda värderingsmetoderna för att jämföra olika metoder för att hitta nyckelord.

4.7.1 Implementation

Implementationen av denna modul använder sig av spaCy som är ett bibliotek med öppen källkod som används för NLP [8]. Med hjälp av dess dokumentation om hur nya modeller kan tränas utformades modulen för att kunna träna ett nytt nätverk som kan hitta ord och stycken som tillhör andra kategorier än de som tidigare varit implementerade i spaCy [22]. Genom att använda sig av träningsdatan som Generator.py genererar tränades en modell för att kunna hitta ord tillhörande kategorin ”SKILL”, vilket defineras som tekniska färdigheter. Denna modul sparar även den tränade modellen för senare användning och gör det möjligt att återuppta träningen av modellen i ett senare skede.

4.8 Utils.py

Python är ett mycket kraftfullt språk för att behandla stora mängder data med ett flertal färdiga funktioner för detta ändamål. Dock kom det att krävas något mer specifika metoder för att behandla den typen av data som behövdes bearbetas. I samband med detta uppkom behovet av en klass med utökad funktionalitet för behandling av vår data.

Utilities innehåller metoder för att hantera saker som att:

- Skapa sträng-representationer av datastrukturer såsom dictionaries (kapitel 4.5.3)
- Skrivning av datastrukturer till filer
- Skapa filhierarkin som krävs för att köra programmet
- Formatera om konsultprofiler från .xlsx till .txt
- Hämta nyckelord ur en fil med hjälp av preprocessorn
- Köra samtliga konsultprofiler och annonser genom preprocessorn
- Matcha samtliga konsultprofiler mot samtliga annonser med samtliga matchningsalgoritmer
- Ranka matchningsresultat så att endast de bästa kandidaterna kommer med i resultatet

Med hjälp av funktionaliteten i Utils-modulen kunde även projekets resultat produceras i form av tabeller (Se kapitel 5 samt bilaga A och B). Detta skedde med två olika metoder som arbetar tillsammans för att förbehandla och matcha alla filer.

5 Resultat

I detta kapitel presenteras projektets resultat i form av åtta olika tabeller som togs fram baserat på tio olika uppdragsannonser matchade mot 136 olika konsultprofiler. Detta gjordes med hjälp utav det framtagna systemet som baserades på systemkonstruktionen beskriven i kaptel 4. Dessa återfinns i bilaga A och B. Anledningen till att de är uppdelade i två olika bilagor är på grund av att alla resultat i bilaga A använder sig av matchingsmetoder som kräver viktade nyckelord och de resultat som finns bilaga B kräver inte det. Varje tabell är märkt med en siffra 1 till 4 och refereras till som exempelvis A3, B2, etc. Varje post i tabellen är på samma format som i exemplet i kapitel 4 och varje tabells rubrik är på formatet:

Värderingsmetod för konsultdokumentet - Matchningsmetod - Värderingsmetod för annonsen

De olika värderingsmetoderna är de som beskrivits tidigare i kapitel 4.4 och 4.7:

PRE - Dokumentet körs endast genom preprocessorn, inga vikter tilldelas
TextRank - Nyckelord värderas baserat på TextRank
Tf-idf - Nyckelord värderas baserat på Tf-idf
NER - Nyckelord tas fram baserat på om de känns igen av NER-nätverket

De olika matchingsmetoderna är de som beskrivits tidigare i kapitel 4.5:

Subset - Matcha enligt metoden beskriven i 4.5.1
Jaccard - Matcha enligt metoden beskriven i 4.5.2
Cosine - Matcha enligt metoden beskriven i 4.5.3

För varje annons har endast de fem bästa resultaten plockats ut för att göra datan mer överskådlig samt för att sålla ut de sämre resultaten. Av samma anledning valdes även endast tio annonser ut, samtliga annonser är från IT- och teknikindustrin för att reflektera företagets målmarknad.

De annonser som valts ut har följande rubriker och stämmer överens med numreringen i bilagorna:

1. *Embedded Developer*
2. *Senior Java Developer - AWS*
3. *Business Analyst Senior FCIA Strategic Design*
4. *Application Developer - Embedded SW/C/C++*
5. *Sales Proposal Manager (RFS coordinator)*
6. *Solution Architect Integration and API's*
7. *Development engineer, Electrics- and Cable Harness design*
8. *Azure DevOps engineer*
9. *Controller/Business Navigator*
10. *Technical Integrator in projects*

Dessa kommer att användas för att enklare kunna diskutera olika aspekter av kvalitén hos resultaten.

6 Diskussion

I detta kapitel diskuteras de resultat som systemet har producerat. Kapitlet kommer även att diskutera svårigheterna med att utveckla denna typ av system och vad det finns för möjligheter att fortsätta utveckla i det nuvarande systemet.

Med skäl att skydda känsliga och personliga uppgifter kan dessvärre inga konsultprofiler eller uppdragsannonser presenteras i sin helhet. Dessa kommer att beskrivas mer anonymt genom kapitlet.

De resultat som diskuteras avser enbart resultaten i bilaga A och B. Det kan argumenteras för att en större eller mindre resultatmängd skulle kunna påvisa andra slutsatser än de som diskuteras här. Den huvudsakliga anledningen till att de utvalda resultaten inte är fler är på grund av tiden som skulle krävas för att bedöma dess pålitlighet. Anledning till att de inte är färre är för att försöka undvika bekräftelseförväntning.

6.1 Pålitligheten hos resultatet

Genom att granska resultaten i bilaga A och B så är det möjligt att avgöra hur bra systemet anser att en viss matchning är jämfört med hur bra en subjektiv bedömning anser att samma resultat är. Vad som avses med en subjektiv bedömning är hur projektarbetes gruppmedlemmar har bedömt lämpligheten hos en konsult gentemot ett visst uppdrag. Detta baserat på kraven i uppdraget och konsultens kunskap och erfarenhet, alltså likt hur en rekryterare hos företaget arbetar. Med detta är det då möjligt att till viss grad avgöra hur pålitliga resultaten är.

För att enkelt få en tydlig indikation på vilken kandidat systemet ansett vara den bästa för ett visst uppdrag så har ett medelvärde av resultaten använts, detta medelvärde är inte en del av systemets funktionalitet utan något som tagits fram genom att gå igenom resultaten i bilagorna. I de flesta fall har endast de två eller tre bästa kandidaterna undersökts för respektive uppdrag. Exempelvis så har annons 1 i bilaga A fyra olika kolumner, konsult 240 har valts ut som bäst lämpad för uppdraget i fråga i samtliga kolumner. Detta kan ses som en tydlig indikation på vad systemet anser vara en lämplig kandidat.

Det verkar finnas en viss tillförlitlighet i systemet då flera av resultaten är passande matchningar i den mening att konsulten i fråga innehar samtliga eller de flesta av egenskaperna som efterfrågats i annonsen. Många av resultaten ligger dock i en gråzon på så vis att kandidaten möjligtvis skulle kunna vara lämplig men det är svårt att göra en klar bedömning då vissa egenskaper saknas hos kandidaten som i vissa fall kan ses som underförstådda eller snarlika de som efterfrågas. Sedan finns det även några resultat som är helt irrelevanta i den mening att matchningen inte går ihop på något vis. Exempelvis så saknas all kunskap som efterfrågas i annonsen eller att konsulten jobbar inom ett helt annat område.

6.1.1 Granskning av resultat

Resultaten i bilaga A är som tidigare nämnt producerade med hjälp utav systemets viktade matchningalgoritmer medan resultaten i bilaga B är producerade med hjälp av systemets oviktade matchningsalgoritmer. Något som kan ses som positivt är att de viktade respeptive oviktade algoritmerna oftast ger liknande bäst lämpade kandidater vilket kan vara en starkare indikation att kandidaterna faktiskt är de bäst lämpade kandidaterna baserat på de befintliga kandidaterna. Detta är dock inte alltid något bra då det kan ge upphov till falska positiva fall.

Positiva resultat

Dessa är resultaten där systemet har bedömt en kandidat som lämplig och där resultatet vid närmare granskning har visat sig vara en bra matchning. De kandidater som har fått de mest pålitliga resultaten är följande:

Bilaga A		Bilaga B	
Annons:	Konsult:	Annons:	Konsult:
1	240, 290	1	290
4	200, 240	4	200, 112
7	417, 329	9	144, 114

Dessa resultat kan ses som kandidater som med hög sannolikhet passar in på respektive uppdrag och kan bidra till en drastiskt förenklad rekryteringsprocess med mindre krav på kontroll från rekryterarens sida.

Gråzonen

Om man ser till resultaten som ligger i mer av en gråzon är dessa som tidigare nämnt svårare att göra en bra bedömning av. Exempelvis uppfylls bara en del av de kraven som efterfrågas i en annons eller så har konsulten snarlika kunskaper men inte exakt de som efterfrågas. De kandidater som ligger i gråzonen är följande:

Bilaga A		Bilaga B	
Annons:	Konsult:	Annons:	Konsult:
1	491	1	491
2	264	2	248, 94
4	290	3	125
6	382	5	116
7	315	6	382, 248
8	161, 494	7	315, 336
		8	115
		10	246

Dessa resultat var som sagt svårare att göra en klar bedömning av. De kan dock vara en tillräckligt god indikation för att ge en rekryterare vid företaget en bra startpunkt om vilka kandidater som är värda att titta närmare på för att minska tiden som krävs för att välja ut en lämplig kandidat.

Falska positiva fall

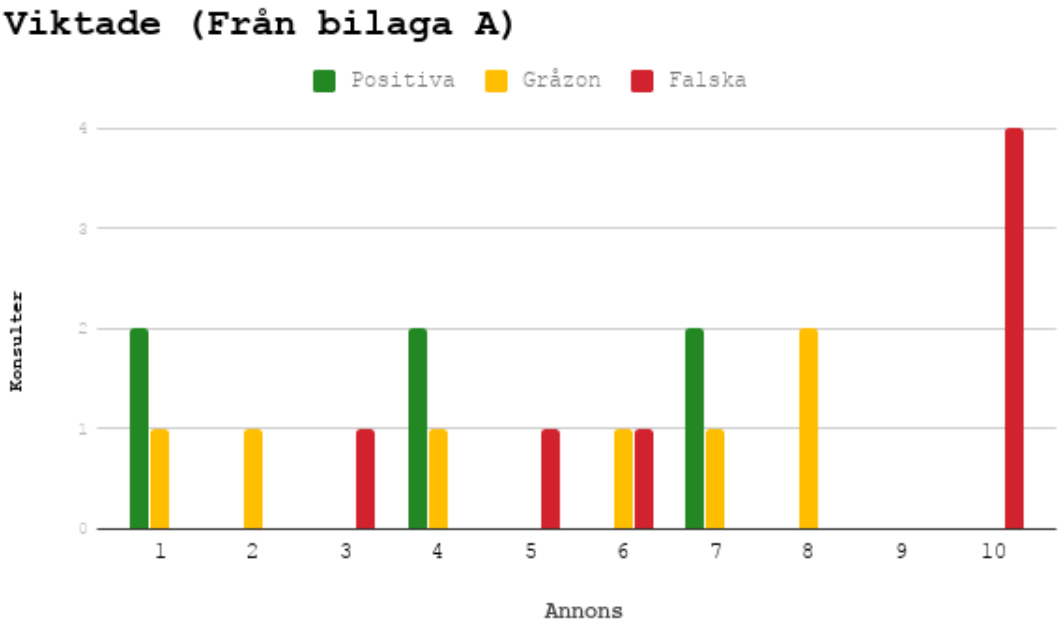
Till sist har systemet även producerat några mer eller mindre oanvändbara eller missledande resultat. Exempelvis där systemet har rankat en kandidat väldigt högt men matchningen är väldigt dålig när man tittar närmare på vad som efterfrågas och konsultens egenskaper. De kandidater som dykt upp som missvisande är följande:

Bilaga A		Bilaga B	
Annons:	Konsult:	Annons:	Konsult:
3	144	3	318
5	416	5	118
6	297	8	158
10	273, 274, 297, 197	10	273

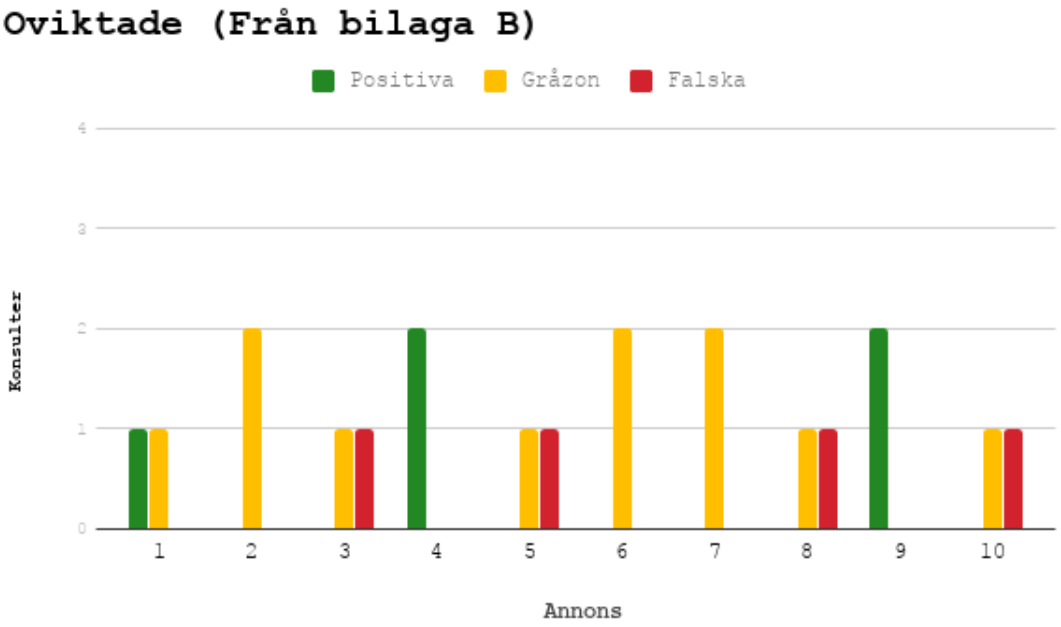
Dessa resultat är alla falska positiva fall där systemet anser att kandidaten är bra men vid närmare granskning finns det inget som ligger till grund för att så skulle vara fallet.

Summering av granskningen

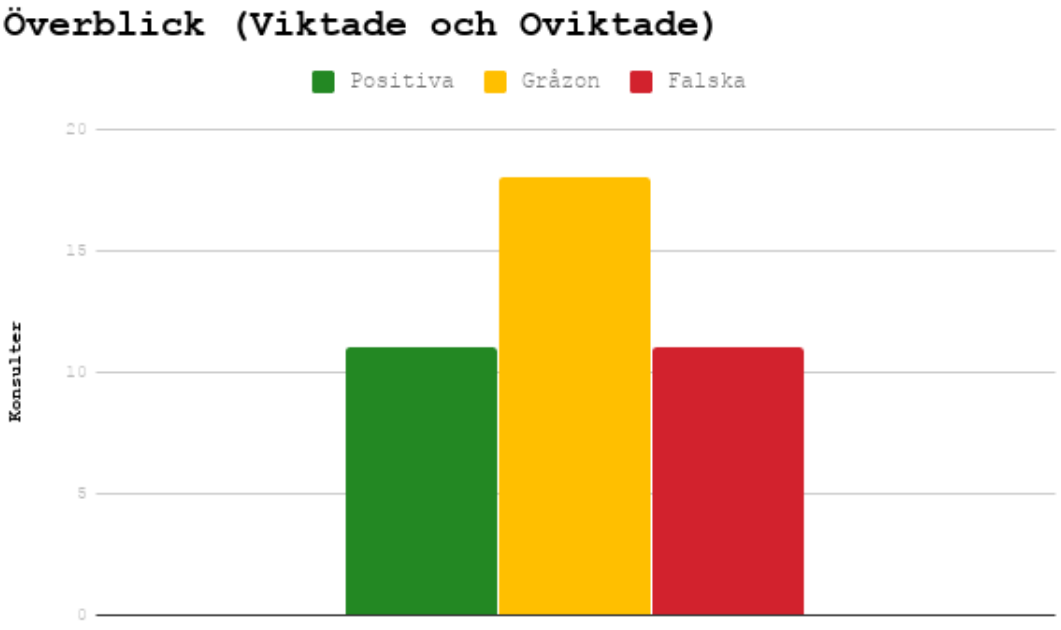
Nedan är figurer som summerar antalet positiva-, gråzons- och falska positiva fall från granskningen av resultaten för varje annons på ett mer överskådligt vis.



Figur 6.1: Summering av granskning av bilaga A



Figur 6.2: Summering av granskning av bilaga B



Figur 6.3: Överblick av de totala antalet från granskning av bilagor A och B

Utifrån figur 6.3 kan vi se att det är lika många positiva som falska positiva fall och att resultaten till större del består av fall i gråzonen som behöver granskas noggrannare för att avgöra om de är bra matchningar eller inte.

6.1.2 Faktorer som påverkar pålitligheten

Efter att ha granskat resultaten kunde några samband upptäckas mellan vilka matchningar som var bra i förhållande till hur annonsen var utformad. Ett av de mer tydliga sambanden var hur en annons efterfrågar hårda- respektiva mjuka egenskaper hos en kandidat.

Annons 1 och 4 har gemensamt att de är båda relativt korta annonser som efterfrågar *hårda* egenskaper, alltså egenskaper som är enkla att beskriva konkret, till exempel CAD, Java, SAP etc. Alla kandidaterna som visat sig lämpliga har haft samtliga eller många av de efterfrågade egenskaperna i dessa annonser.

Annons 5, 9 och 10 skiljer sig från dessa och efterfrågar fler *mjuka* färdigheter, såsom ledarskap, kommunikationsförmåga och dylikt. Det efterfrågas dock även hårda färdigheter utöver dessa såsom krav på kunskap inom affärsanalys. Dessa gav upphov till många av resultaten som hamnade i gråzonen.

Av de resultat som dök upp som falska positiva fall var det ofta så att annonsen var lång och efterfrågade huvudsakligen mjuka egenskaper, vilket kan ge bra utslag från många av algoritmerna, men ofta på ett missvisande sätt. Till exempel ordet ”communication” kan vara en mjuk egenskap, men kan också vara ett resultat av hur vi ignorerat att hantera multiwords [20] där det kan vara en del av andra ord som exempelvis ”satellite communication” vilket är en tydligt hård egenskap jämfört med bara ”communication”. Detta gäller även för hur konsultprofilerna är utformade.

En annan tydlig faktor verkar vara längden på annonserna. Överlag verkar längre annonser ge sämre matchningar än kortare. Det kan bero på att de längre annonserna ofta innehåller mycket irrelevant information för att skapa en bra matchning, som till exempel beskrivning av företaget som annonsen kommer från eller om den innehåller mycket kontaktuppgifter eller dylikt.

6.2 Vidareutveckling

Efter färdigställandet av systemet och analys av resultaten dök det upp ett antal funderingar på hur systemet kan vidareutvecklas. Ett flertal idéer som kan vara intressanta att arbeta vidare med för att göra systemet mer pålitligt och användarvänligt tas upp i denna sektion.

6.2.1 Användarvänlighet

Då systemets användarvänlighet låg utanför projektets ramar så finns det ett enormt utrymme för utveckling på just den fronten. I dagsläget måste en som förstår programmering i Python använda sig av funktioner i main.py för att få fram de efterfrågade resultaten. Detta är inte speciellt användarvänligt och hindrar de flesta från att använda systemet överhuvudtaget.

För att göra systemet användbart för flera så skulle ett simpelt kommandotolksverktyg kunna utvecklas som gör det möjligt att välja vilka resultat systemet ska skriva ut. Detta kan vara saker som de fem bästa kandidaterna för just en annons, de bästa kandidaterna för flera annonser, en kandidats bästa matchning med en annons med mera. Alternativt kan ett grafiskt användargränssnitt utvecklas som skulle kunna göra det mycket enklare att använda för de rekryterare som inte har erfarenhet av kommandotolksverktyg.

6.2.2 CosineMatch

Att använda cosinus-likhet för att skapa bra matchningar visade sig vara mycket användbart och gav i många fall flera bra resultat. Det finns dock tydliga nackdelar med den implementation som gjordes i systemet. Det största problemet är om en kandidat har kunskaper utöver vad som krävs i en annons. Detta är inte något som borde bidra till en sämre matchning utan tvärtom. Detta kan demonstreras med ett exempel:

```
Annons:
Java, C++, Python

Kandidat 1:          Kandidat 2:
Java, C++            Java, C++, Python, SQL, Lego, Scratch, Scrum
```

Kandidat 2 verkar vara mycket bättre lämpad, men såsom CosineMatch har implementerats så kommer den personen få ett sämre resultat på grund av hur vektorerna bildas för att göra matchningen, ett exempel på hur detta sker finns i kapitel 2.8.2. För att motverka detta skulle implementationen behöva ses över för att ta hänsyn till detta.

6.2.3 Matchningsresultat

Resultaten som systemet producerar skulle behöva göras mer överskådliga. Ett förslag är att producera resultaten som en enda tabell istället för åtta olika som sedan måste tolkas manuellt. Nästa fråga som uppstår är då hur denna ensamma tabell ska konstrueras.

Ett förslag är att ta ett medelvärde av olika algoritmer på samma vis som gjordes när resultaten analyserades. En annan metod skulle kunna vara att lägga fokus på att vidareutveckla en enda algoritm och enbart basera resultaten på den.

6.2.4 Hämtning av uppdragsannonser

I systemets slutliga tillstånd hämtas endast uppdragsannonser från en källa. Detta kan utvecklas och det kan enkelt integreras flera källor att hämta annonser från som till exempel andra hemsidor eller interna databaser.

6.3 Miljö- och etiska aspekter

Förutsatt att systemet tas i bruk skulle detta kunna leda till minskad stress hos medarbetare på företaget och därmed leda till ökad trivsel och bättre förutsättningar på arbetsplatsen. Det kan även leda till ökad lönsamhet då effektivteten hos företaget kan gå upp i samband med att systemet tas i bruk. En mindre aspekt skulle även kunna vara en minskad förbrukning av papper till följd av att stora delar av rekryteringsprocessen digitaliseras med hjälp av detta system.

7 Frågeställningar och svar

I detta kapitel tas svar baserat på projektets frågeställningar från kaptitel 1.4 upp. Varje fråga behandlas i ett eget avsnitt som diskuterar bland annat för- och nackdelar med olika metoder som använts, alternativa användningsområden och argument för några av projektets designval. Flera utav frågorna fick sina svar redan under systemets konstruktion då de behövde besvaras i samband med utformningen av vissa moduler, därav blir några utav svaren till största del en referens till sina respektiva moduler i kapitel 4.

7.1 Hur kvantifieras systemets effektivitet?

För att kvantifiera systemets effektivitet är det möjligt att se på dess tidsefftivitet. Just en av de positiva egenskaperna hos systemet är dess tidseffektivitet. Som ett exempel kan systemet hämta ner ett hundratal olika uppdragsannonser, preprocessa samtliga av dessa tillsammans med 136 konsultprofiler samt utföra matchningsprocessen för samtliga dokument med åtta olika kombinationer på några få minuter, vilket ger upphov till 217 600 resultatposter för 200 annonser. Då en av tankarna med ett färdigställt system är att kunna köra programmet i sin helhet en gång per dygn för att få fram matchningsresultat så klarar systemet av detta mycket bra rent tidsmässigt och eliminerar behovet av stora mängder datorkraft.

7.2 Vilka programspråk och bibliotek lämpar sig bäst för att utforma applikationen?

När det kommer till att arbeta med flytande text, natural language processing och databehandling så har Python visat sig vara ett mycket bra verktyg för detta. Det finns en enorm tillgång på färdiga bibliotek och lösningar för många av de problem som stöttes på under projektets gång. Några av de mest nämnvärda python-biblioteken var:

- NLTK
- spaCy
- NumPy
- SciPy
- NetworkX
- BeautifulSoup

Alla dessa möjliggjorde en mycket effektiv utvecklingsmiljö där prototyper snabbt kunde utvecklas och testas. För att möjliggöra projektets slutförande inom dess tidsram ansågs Python vara ett mycket kritiskt verktyg för att klara av detta. Värt att nämna är även ramverket Selenium som visade sig vara väldigt användbart för webskrapning av dynamiska element på webbplatser.

7.3 Hur ska data från databasen och annonser förbehandlas?

Sättet som valdes för att förbehandla data var med hjälp utav preproccessormodulen som beskrevs i kapitel 4.3. En stor fördel med denna komponent är att den är relativt generaliserad. Så länge dokumentet den ska förbehandla är på engelska så klarar den av mer eller mindre vilken sorts text som helst. Man skulle lika gärna kunna förbehandla till exempel ett kapitel ur en vetenskaplig text för att sedan plocka ut nyckelord ur den på samma sätt som ur uppdragsannonser.

7.4 På vilket sätt ska matchningsprocessen ske?

För att hantera matchning mellan konsultprofiler och uppdragsannonser användess tre olika metoder. Dessa var jaccard-, subset- och cosine match. Anledning till att ha tre olika metoder var för att kunna jämföra de olika resultaten med varandra för att avgöra om någon av metoderna stack ut på något sätt. Den mest intressanta faktorn att ta hänsyn till var resultatens pålitlighet, det vill säga huruvida en matchning faktiskt är bra eller inte baserat på procentsatsen som producerats av matchningsmetoden. Detta diskuteras mer ingående i nästa fråga.

7.5 Vad räknas som en bra matchning?

När det kommer till att avgöra vad som menas med en bra matchning så är det en relativt subjektiv fråga. Generellt sett kan man dock säga att en bra matchning är där en konsultprofil har färdigheter som reflekterar de efterfrågade färdigheterna i den givna annonsen. Från de givna resultaten kan det dock vara svårt att avgöra hur många och just vilka färdigheter som en konsultprofil har uppfyllt, och just hur viktiga dessa är för uppdraget.

För att avgöra om en matchning är bra krävs det i slutändan ett manuellt beslut och tanken är att resultaten kan hjälpa till med att ta fram förslag om vilka konsulter som kan vara värda att studera noggrannare för en rekryterare vid företaget.

7.6 Vilka metoder finns för att bearbeta löpande text som annonser ofta består av?

I stora drag så används Natural language processing för att bearbeta löpande text. Inom NLP så finns det dock många metoder och tekniker för att bearbeta löpande text men de som inte har nämnts i tidigare kapitel är metoder som inte har visat sig vara användbara eller metoder som sträcker sig utanför projektets ramar.

De metoder som visade sig vara mest användbara i projektet var POS-tagging, lemmatisering, filtrering av stopwords och NER.

7.7 Bör artificiell intelligens tillämpas?

Flera av designvalen som uppstod under utformningen av systemet kom att kräva tillämpningar av AI, framförallt vad gäller just användning av NLP. Detta då NLP är ett subfält till just AI vilket gör det ganska oundvikligt att inte använda det om man inte istället väljer ett helt annat tillvägagångssätt för att lösa många av problemen som uppstår i samband med behandling av löpande text.

7.8 Vad begränsar systemets skalbarhet?

Huvudsakligen så är systemet begränsat av beräkningskraft då det krävs ytterligare beräkningar för varje konsult och annons som läggs till i systemet vilket leder till längre beräkningtider. Detta leder till att systemet kan bli oanvändbart vid allt för stora mängder konsulter och annonser då beräkningstiden kan bli enorm.

8 Slutsats

Målet med projektet var att skapa ett system för att automatisera delar av matchningsprocessen mellan företagets konsulter och uppdrag. Projektet har nått detta mål och systemet producerar matchningsresultat som kan användas av rekryterare som förslag på en matchning mellan en konsult och ett uppdrag. Flera av dessa matchningsresultat har visat sig vara pålitliga och kan användas som bra riktlinjer för att hitta lämpliga konsulter till olika uppdrag. Systemet har dock även producerat missvisande resultat i form av falska positiva fall. Överlag verkar dock resultaten som är bra eller ligger i en gråzon vara fler än de som är missvisande.

Det finns mycket utrymme för utveckling av systemet i form av användarvänlighet, sammanställning av resultat och framförallt pålitligheten hos resultaten. I nuläget rekommenderas det att systemets resultat används som riktlinjer för att välja ut lämpliga kandidater till olika uppdrag av en rekryterare. Det krävs alltså fortfarande en del manuellt arbete även om delar av processen har kunnat automatiseras. Detta ses som ett stort framsteg gentemot hur rekryteringsprocessen ser ut i dagsläget i den mening att mängden arbete som krävs för att hitta en lämplig kandidat kan minskas drastiskt.

Referenser

[1] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[2] P. M. Nadkarni, L. Ohno-Machado och W. W. Chapman, ”Natural language processing: an introduction”, *Journal of the American Medical Informatics Association*, årgång 18, nummer 5, sidorna 544–551, sept. 2011, ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464. webbadress: https://doi.org/10.1136/amiajnl-2011-000464.

[3] Python Software Foundation. (n.d.). Welcome to python.org, webbadress: Python.org (senast hämtad 2019-04-25).

[4] S. Bird, E. Klein och E. Loper, *Natural Language Processing with Python*. O’Reilly Media, Inc., 2009.

[5] Tfidf.com. (n.d.). Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining, webbadress: www.tfidf.com (senast hämtad 2019-04-25).

[6] R. Mihalcea och P. Tarau. (n.d.). TextRank: Bringing Order into Texts, webbadress: https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf (senast hämtad 2019-04-25).

[7] D. Nadeau och S. Sekine, ”A survey of named entity recognition and classification”, *Lingvisticae Investigationes*, årgång 30, nummer 1, sidorna 3–26, 2007.

[8] Explosion AI. (n.d.). spaCy, Industrial-Strength Natural Language Processing, webbadress: https://spacy.io// (senast hämtad 2019-05-02).

[9] Wikipedia. (n.d.). Web scraping, webbadress: https://en.wikipedia.org/wiki/Web_scraping (senast hämtad 2019-04-25).

[10] ——, (n.d.). Cosine similarity, webbadress: https://en.wikipedia.org/wiki/Cosine_similarity (senast hämtad 2019-04-25).

[11] Scrum.org. (n.d.). What is Scrum?, webbadress: https://www.scrum.org/resources/what-is-scrum (senast hämtad 2019-04-25).

[12] Atlassian. (n.d.). Trello, webbadress: https://trello.com/ (senast hämtad 2019-04-25).

[13] GitHub Inc. (n.d.). GitHub, webbadress: https://github.com/ (senast hämtad 2019-04-25).

[14] V. Driessen, ”A successful Git branching model”, 2010. webbadress: https://nvie.com/posts/a-successful-git-branching-model/ (senast hämtad 2019-04-25).

[15] L. Richardson. (n.d.). Beautiful Soup, webbadress: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (senast hämtad 2019-04-25).

[16] Mozilla. (n.d.). JavaScript, webbadress: https://developer.mozilla.org/en-US/docs/Web/JavaScript (senast hämtad 2019-04-25).

[17] Seleniumhq.org. (n.d.). Selenium, webbadress: https://www.seleniumhq.org/ (senast hämtad 2019-04-25).

[18] Wikipedia. (n.d.). Regular expression, webbadress: https://en.wikipedia.org/wiki/Regular_expression (senast hämtad 2019-04-25).

[19] N. developers. (n.d.). NetworkX, webbadress: https://networkx.github.io/ (senast hämtad 2019-04-25).

[20] J. Legrand och R. Collobert, ”Phrase Representations for Multiword Expressions”, 2016. webbadress: https://www.aclweb.org/anthology/W16-1810/ (senast hämtad 2019-04-25).

[21] Wikipedia. (n.d.). Jaccard index, webbadress: https://en.wikipedia.org/wiki/Jaccard_index (senast hämtad 2019-05-03).

[22] Explosion AI. (n.d.). Training spaCy’s Statistical Models, webbadress: https://spacy.io/usage/training (senast hämtad 2019-05-02).

A CosineMatch

```
1. TextRank - Cosine - TextRank

Advert: 1

Consultant:  240, Match: 30.0%
Consultant:  276, Match: 26.0%
Consultant:  491, Match: 26.0%
Consultant:  206, Match: 24.0%
Consultant:  115, Match: 23.0%

Advert: 2

Consultant:  242, Match: 16.0%
Consultant:  248, Match: 16.0%
Consultant:  264, Match: 16.0%
Consultant:  345, Match: 16.0%
Consultant:   94, Match: 16.0%

Advert: 3

Consultant:  144, Match: 22.0%
Consultant:  125, Match: 19.0%
Consultant:  173, Match: 17.0%
Consultant:  322, Match: 17.0%
Consultant:  300, Match: 16.0%

Advert: 4

Consultant:  200, Match: 30.0%
Consultant:  238, Match: 23.0%
Consultant:  240, Match: 22.0%
Consultant:   65, Match: 22.0%
Consultant:  112, Match: 21.0%

Advert: 5

Consultant:  416, Match: 14.0%
Consultant:  116, Match: 12.0%
Consultant:  144, Match: 10.0%
Consultant:  398, Match: 10.0%
Consultant:   63, Match:  9.0%

Advert: 6

Consultant:  297, Match: 20.0%
Consultant:  382, Match: 18.0%
Consultant:  116, Match: 17.0%
Consultant:   94, Match: 17.0%
Consultant:  273, Match: 16.0%

Advert: 7

Consultant:  329, Match: 24.0%
Consultant:  315, Match: 21.0%
Consultant:  417, Match: 21.0%
Consultant:  109, Match: 20.0%
Consultant:  125, Match: 20.0%

Advert: 8

Consultant:  494, Match: 19.0%
Consultant:  161, Match: 18.0%
Consultant:  258, Match: 17.0%
Consultant:   94, Match: 16.0%
Consultant:  380, Match: 15.0%

Advert: 9

Consultant:   86, Match: 10.0%
Consultant:  398, Match:  9.0%
Consultant:   94, Match:  9.0%
Consultant:  144, Match:  8.0%
Consultant:  410, Match:  8.0%

Advert: 10

Consultant:  273, Match: 42.0%
Consultant:  196, Match: 40.0%
Consultant:  274, Match: 35.0%
Consultant:  240, Match: 32.0%
Consultant:  297, Match: 31.0%
```

```
2. TextRank - Cosine - Tf-idf

Advert: 1

Consultant:  240, Match: 21.0%
Consultant:  290, Match: 20.0%
Consultant:  249, Match: 19.0%
Consultant:  133, Match: 16.0%
Consultant:  200, Match: 16.0%

Advert: 2

Consultant:  248, Match: 12.0%
Consultant:  416, Match: 11.0%
Consultant:  206, Match: 10.0%
Consultant:  264, Match: 10.0%
Consultant:  379, Match: 10.0%

Advert: 3

Consultant:  144, Match: 13.0%
Consultant:  173, Match: 11.0%
Consultant:  125, Match: 10.0%
Consultant:  159, Match: 10.0%
Consultant:  300, Match:  9.0%

Advert: 4

Consultant:  200, Match: 49.0%
Consultant:  240, Match: 39.0%
Consultant:  249, Match: 33.0%
Consultant:  290, Match: 33.0%
Consultant:  133, Match: 30.0%

Advert: 5

Consultant:  416, Match: 12.0%
Consultant:  116, Match: 10.0%
Consultant:  414, Match:  7.0%
Consultant:  208, Match:  6.0%
Consultant:  117, Match:  5.0%

Advert: 6

Consultant:  382, Match: 24.0%
Consultant:  494, Match: 14.0%
Consultant:  116, Match: 13.0%
Consultant:  347, Match: 13.0%
Consultant:  379, Match: 13.0%

Advert: 7

Consultant:  417, Match: 24.0%
Consultant:  329, Match: 15.0%
Consultant:  109, Match: 13.0%
Consultant:  301, Match: 13.0%
Consultant:  315, Match: 13.0%

Advert: 8

Consultant:  161, Match: 24.0%
Consultant:  494, Match: 20.0%
Consultant:  258, Match: 18.0%
Consultant:  380, Match: 16.0%
Consultant:  112, Match: 14.0%

Advert: 9

Consultant:   86, Match:  9.0%
Consultant:  398, Match:  6.0%
Consultant:  410, Match:  6.0%
Consultant:  101, Match:  4.0%
Consultant:  119, Match:  4.0%

Advert: 10

Consultant:  273, Match: 29.0%
Consultant:  274, Match: 23.0%
Consultant:  297, Match: 21.0%
Consultant:  196, Match: 19.0%
Consultant:  132, Match: 18.0%
```

```
3. Tf-idf - Cosine - Tf-idf

Advert: 1

Consultant:  240, Match: 22.0%
Consultant:  290, Match: 19.0%
Consultant:  491, Match: 16.0%
Consultant:  202, Match: 14.0%
Consultant:  249, Match: 14.0%

Advert: 2

Consultant:  416, Match: 11.0%
Consultant:  248, Match: 10.0%
Consultant:  264, Match:  9.0%
Consultant:   65, Match:  8.0%
Consultant:  206, Match:  7.0%

Advert: 3

Consultant:  144, Match: 12.0%
Consultant:  117, Match: 10.0%
Consultant:  159, Match:  9.0%
Consultant:  173, Match:  9.0%
Consultant:  398, Match:  7.0%

Advert: 4

Consultant:  240, Match: 39.0%
Consultant:  200, Match: 37.0%
Consultant:  290, Match: 29.0%
Consultant:  205, Match: 23.0%
Consultant:  249, Match: 23.0%

Advert: 5

Consultant:  116, Match: 10.0%
Consultant:  416, Match: 10.0%
Consultant:  414, Match:  8.0%
Consultant:  125, Match:  5.0%
Consultant:  117, Match:  4.0%

Advert: 6

Consultant:  382, Match: 32.0%
Consultant:  494, Match: 18.0%
Consultant:   61, Match: 14.0%
Consultant:  297, Match: 10.0%
Consultant:  248, Match:  8.0%

Advert: 7

Consultant:  417, Match: 23.0%
Consultant:  315, Match: 11.0%
Consultant:  329, Match: 11.0%
Consultant:  109, Match:  8.0%
Consultant:  132, Match:  8.0%

Advert: 8

Consultant:  161, Match: 32.0%
Consultant:  494, Match: 22.0%
Consultant:  258, Match: 18.0%
Consultant:  380, Match: 17.0%
Consultant:  238, Match: 16.0%

Advert: 9

Consultant:   86, Match: 14.0%
Consultant:  398, Match:  9.0%
Consultant:  129, Match:  5.0%
Consultant:  160, Match:  5.0%
Consultant:  173, Match:  5.0%

Advert: 10

Consultant:  273, Match: 26.0%
Consultant:  132, Match: 18.0%
Consultant:  322, Match: 18.0%
Consultant:  274, Match: 16.0%
Consultant:  267, Match: 13.0%
```

```
4. Tf-idf - Cosine - TextRank

Advert: 1

Consultant:  240, Match: 23.0%
Consultant:  491, Match: 20.0%
Consultant:  115, Match: 17.0%
Consultant:  290, Match: 17.0%
Consultant:  273, Match: 16.0%

Advert: 2

Consultant:  345, Match: 11.0%
Consultant:  115, Match: 10.0%
Consultant:   94, Match: 10.0%
Consultant:  206, Match:  9.0%
Consultant:  248, Match:  9.0%

Advert: 3

Consultant:  144, Match: 21.0%
Consultant:  173, Match: 13.0%
Consultant:  398, Match: 13.0%
Consultant:   94, Match: 13.0%
Consultant:  159, Match: 12.0%

Advert: 4

Consultant:  200, Match: 17.0%
Consultant:  240, Match: 17.0%
Consultant:  290, Match: 13.0%
Consultant:   65, Match: 13.0%
Consultant:  112, Match: 12.0%

Advert: 5

Consultant:  416, Match: 12.0%
Consultant:  116, Match: 10.0%
Consultant:   63, Match:  8.0%
Consultant:  144, Match:  7.0%
Consultant:  414, Match:  7.0%

Advert: 6

Consultant:  382, Match: 19.0%
Consultant:  297, Match: 18.0%
Consultant:  494, Match: 14.0%
Consultant:  144, Match: 13.0%
Consultant:   94, Match: 11.0%

Advert: 7

Consultant:  417, Match: 15.0%
Consultant:  132, Match: 13.0%
Consultant:  315, Match: 13.0%
Consultant:  329, Match: 12.0%
Consultant:  109, Match:  9.0%

Advert: 8

Consultant:  161, Match: 23.0%
Consultant:  494, Match: 17.0%
Consultant:  380, Match: 15.0%
Consultant:  238, Match: 14.0%
Consultant:  258, Match: 14.0%

Advert: 9

Consultant:   86, Match: 13.0%
Consultant:  398, Match: 10.0%
Consultant:  160, Match:  9.0%
Consultant:   94, Match:  9.0%
Consultant:  144, Match:  8.0%

Advert: 10

Consultant:  273, Match: 32.0%
Consultant:  322, Match: 25.0%
Consultant:  132, Match: 24.0%
Consultant:  274, Match: 19.0%
Consultant:  297, Match: 19.0%
```

B SubsetMatch och JaccardMatch

```
1. PRE - Subset - NER

Advert: 1

Consultant:  290, Match:   30%
Consultant:  491, Match:   30%
Consultant:  240, Match:   26%
Consultant:  273, Match:   26%
Consultant:  276, Match:   26%

Advert: 2

Consultant:  248, Match:   18%
Consultant:  336, Match:   18%
Consultant:   94, Match:   18%
Consultant:  115, Match:   14%
Consultant:  141, Match:   14%

Advert: 3

Consultant:  125, Match:   33%
Consultant:  318, Match:   33%
Consultant:  336, Match:   33%
Consultant:  102, Match:   25%
Consultant:  132, Match:   25%

Advert: 4

Consultant:  112, Match:   36%
Consultant:  200, Match:   36%
Consultant:  141, Match:   29%
Consultant:  205, Match:   29%
Consultant:  238, Match:   29%

Advert: 5

Consultant:  116, Match:   24%
Consultant:  336, Match:   24%
Consultant:  114, Match:   18%
Consultant:  118, Match:   18%
Consultant:  144, Match:   18%

Advert: 6

Consultant:  248, Match:   12%
Consultant:  304, Match:   12%
Consultant:  336, Match:   12%
Consultant:  382, Match:   12%
Consultant:  409, Match:   12%

Advert: 7

Consultant:  336, Match:   32%
Consultant:  315, Match:   26%
Consultant:  132, Match:   21%
Consultant:  197, Match:   21%
Consultant:  240, Match:   21%

Advert: 8

Consultant:  115, Match:   14%
Consultant:  158, Match:   14%
Consultant:  112, Match:   12%
Consultant:  117, Match:   12%
Consultant:  143, Match:   12%

Advert: 9

Consultant:  114, Match:   19%
Consultant:  336, Match:   19%
Consultant:  112, Match:   12%
Consultant:  116, Match:   12%
Consultant:  125, Match:   12%

Advert: 10

Consultant:  273, Match:   27%
Consultant:  246, Match:   23%
Consultant:  491, Match:   23%
Consultant:  132, Match:   19%
Consultant:  197, Match:   19%
```

```
2. PRE - Jaccard - NER

Advert: 1

Consultant:  290, Match:   23%
Consultant:  491, Match:   23%
Consultant:  240, Match:   19%
Consultant:  273, Match:   17%
Consultant:  274, Match:   15%

Advert: 2

Consultant:  248, Match:   13%
Consultant:  242, Match:   11%
Consultant:  264, Match:   11%
Consultant:   94, Match:   11%
Consultant:  115, Match:   10%

Advert: 3

Consultant:  125, Match:   18%
Consultant:  318, Match:   17%
Consultant:   65, Match:   17%
Consultant:  102, Match:   14%
Consultant:  262, Match:   14%

Advert: 4

Consultant:  200, Match:   21%
Consultant:  112, Match:   17%
Consultant:  141, Match:   16%
Consultant:  238, Match:   16%
Consultant:  240, Match:   16%

Advert: 5

Consultant:  144, Match:   12%
Consultant:  116, Match:   11%
Consultant:  242, Match:   11%
Consultant:  159, Match:   10%
Consultant:  160, Match:   10%

Advert: 6

Consultant:  304, Match:   10%
Consultant:  248, Match:    9%
Consultant:  382, Match:    9%
Consultant:  409, Match:    9%
Consultant:   61, Match:    9%

Advert: 7

Consultant:  315, Match:   18%
Consultant:  132, Match:   13%
Consultant:  240, Match:   13%
Consultant:  274, Match:   13%
Consultant:  280, Match:   13%

Advert: 8

Consultant:  115, Match:   11%
Consultant:  158, Match:   11%
Consultant:  143, Match:   10%
Consultant:  258, Match:   10%
Consultant:  380, Match:   10%

Advert: 9

Consultant:  114, Match:   10%
Consultant:  144, Match:    8%
Consultant:  125, Match:    7%
Consultant:  159, Match:    7%
Consultant:  160, Match:    7%

Advert: 10

Consultant:  273, Match:   18%
Consultant:  491, Match:   17%
Consultant:  246, Match:   15%
Consultant:  132, Match:   14%
Consultant:  274, Match:   14%
```

```
3. PRE - Subset - PRE

Advert: 1

Consultant:  240, Match:   10%
Consultant:  491, Match:   10%
Consultant:  276, Match:    9%
Consultant:  290, Match:    9%
Consultant:  133, Match:    8%

Advert: 2

Consultant:   94, Match:    7%
Consultant:  379, Match:    6%
Consultant:  248, Match:    5%
Consultant:  336, Match:    5%
Consultant:  344, Match:    5%

Advert: 3

Consultant:  144, Match:    8%
Consultant:   94, Match:    8%
Consultant:  125, Match:    7%
Consultant:  208, Match:    7%
Consultant:  300, Match:    7%

Advert: 4

Consultant:  200, Match:   12%
Consultant:  112, Match:   10%
Consultant:  249, Match:   10%
Consultant:  345, Match:   10%
Consultant:  379, Match:   10%

Advert: 5

Consultant:  116, Match:    5%
Consultant:  118, Match:    4%
Consultant:  336, Match:    4%
Consultant:   63, Match:    4%
Consultant:   94, Match:    4%

Advert: 6

Consultant:  336, Match:    5%
Consultant:  382, Match:    5%
Consultant:   94, Match:    5%
Consultant:  116, Match:    4%
Consultant:  144, Match:    4%

Advert: 7

Consultant:  336, Match:    8%
Consultant:  132, Match:    6%
Consultant:  298, Match:    6%
Consultant:  315, Match:    6%
Consultant:  329, Match:    6%

Advert: 8

Consultant:  379, Match:    7%
Consultant:   94, Match:    6%
Consultant:  112, Match:    5%
Consultant:  114, Match:    5%
Consultant:  115, Match:    5%

Advert: 9

Consultant:  410, Match:    6%
Consultant:  144, Match:    5%
Consultant:  398, Match:    5%
Consultant:   94, Match:    5%
Consultant:  160, Match:    4%

Advert: 10

Consultant:  273, Match:   10%
Consultant:  129, Match:    7%
Consultant:  246, Match:    7%
Consultant:  336, Match:    7%
Consultant:  379, Match:    7%
```

```
4. PRE - Jaccard - PRE

Advert: 1

Consultant:  240, Match:    9%
Consultant:  491, Match:    9%
Consultant:  290, Match:    8%
Consultant:  133, Match:    7%
Consultant:  202, Match:    7%

Advert: 2

Consultant:   94, Match:    6%
Consultant:  248, Match:    5%
Consultant:  379, Match:    5%
Consultant:  115, Match:    4%
Consultant:  141, Match:    4%

Advert: 3

Consultant:  144, Match:    8%
Consultant:   94, Match:    7%
Consultant:  125, Match:    6%
Consultant:  208, Match:    6%
Consultant:  300, Match:    6%

Advert: 4

Consultant:  200, Match:   11%
Consultant:  112, Match:    8%
Consultant:  249, Match:    8%
Consultant:  345, Match:    8%
Consultant:  379, Match:    8%

Advert: 5

Consultant:  116, Match:    4%
Consultant:  118, Match:    4%
Consultant:   63, Match:    4%
Consultant:   94, Match:    4%
Consultant:  114, Match:    3%

Advert: 6

Consultant:  382, Match:    5%
Consultant:   94, Match:    5%
Consultant:  116, Match:    4%
Consultant:  144, Match:    4%
Consultant:  297, Match:    4%

Advert: 7

Consultant:  315, Match:    6%
Consultant:  336, Match:    6%
Consultant:  132, Match:    5%
Consultant:  240, Match:    5%
Consultant:  298, Match:    5%

Advert: 8

Consultant:  379, Match:    6%
Consultant:  115, Match:    5%
Consultant:  117, Match:    5%
Consultant:  158, Match:    5%
Consultant:  238, Match:    5%

Advert: 9

Consultant:  144, Match:    4%
Consultant:  398, Match:    4%
Consultant:  410, Match:    4%
Consultant:   94, Match:    4%
Consultant:  160, Match:    3%

Advert: 10

Consultant:  273, Match:    9%
Consultant:  129, Match:    6%
Consultant:  132, Match:    6%
Consultant:  245, Match:    6%
Consultant:  246, Match:    6%
```