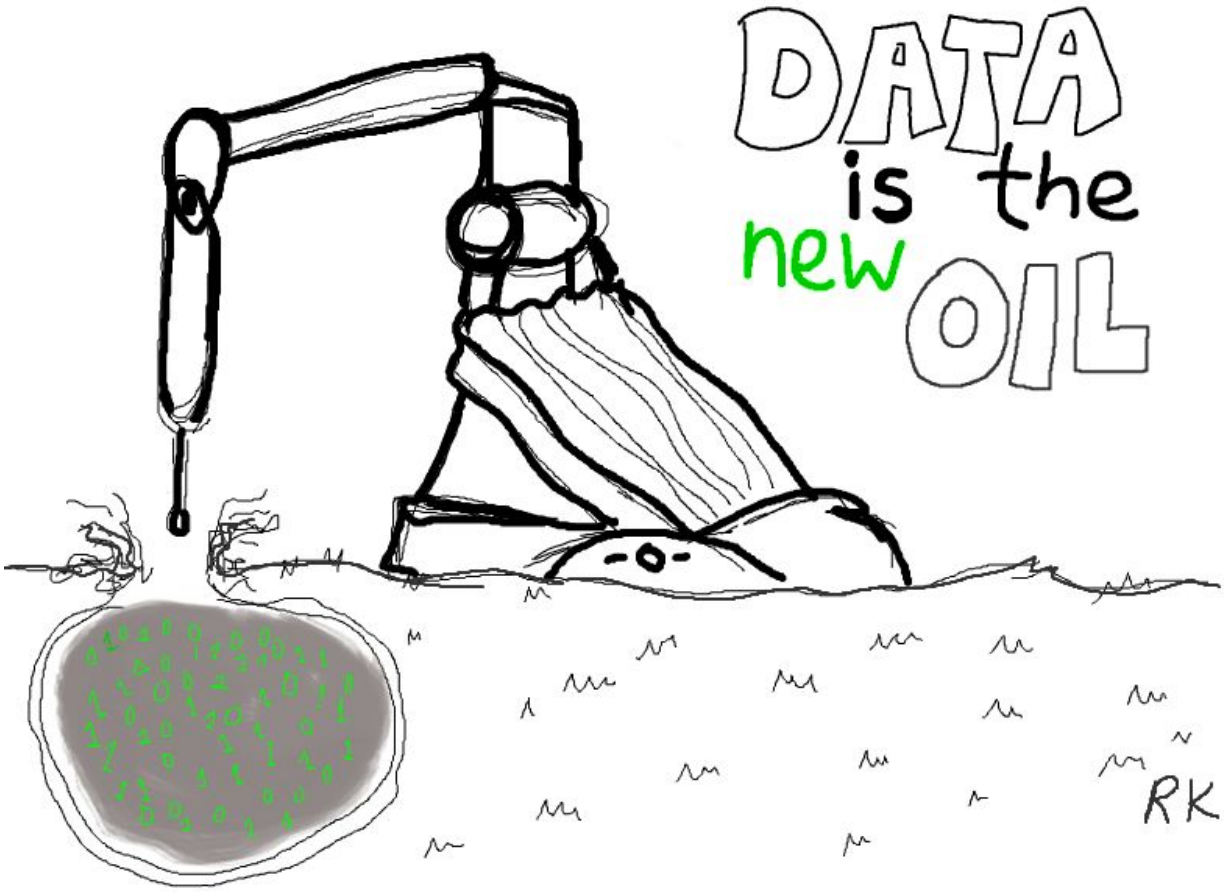
<u>פרוייקט טיוב חיפוש</u>

מאת

<u>יעקב זוארץ</u>

שלום איילון, בוני ולכל קוראי מסמך זה.

<u>השראה למערכת</u>



לפני שאתאר את המערכת והיתרונות המתקבלים מהשימוש בה, אשמח לשתף אתכם במקורות ההשראה והמוטיבציה אשר עמדו לנגד עיני בזמן תיכנון המערכת. בפגישותי עם איילון ובוני נחשפתי לחזון הטכנולוגי של החברה, אשר לכשיתממש יקפיץ את ההביצועים הטכנולוגיים של החברה לדור הבא. ע"פ חזון זה, לחברה תהייה יכולת לחבר בין מחפשי עבודה למישרות באופן מדוייק ביותר עם התאמה גבוהה יותר. ע"פ חזון זה לחברה תהיה טכנולוגיה אשר "לומדת" את המשתמשים שלה ע"י ניתור האינטראקציות שלהם מול המישרות, כמו כן הטכנולוגיה "תלמד" את המישרות וסוגי המועמדים המתאימים לכל משרה. ובסוף הדרך המערכת תשתמש במידע שלמדה באופן אוטומטי כדי להישתפר מיום ליום בהשגת המטרה העליונה של החברה, לחבר בין מחפש עבודה למשרה אשר מתאימה למידותיו,(ו...יש לו סיכוי טוב להיתקבל ע"פ המצב העכשווי של שוק המישרות, זו תוספת שלי :-).

הרעיון הבסיסי

במשך זמן רב ניסיתי לענות על השאלה, מהם המאפיינים הדומים בין מחפש העבודה לבין משרה מוצעת? והאמת, התשובה חמקה ממני עד לנקודה אשר בה הבנתי איפה טעיתי, אין מאפיינים דומים מאחר ושניהם מתארים בדיוק את אותו דבר, **את מחפש העבודה**. אני אסביר, מחפש עבודה מתאר את עצמו בעזרת קורות החיים ובעזרת סוכני החיפוש שעומדים לרשותו, ומהצד השני, המשרה מתארת את מחפש העבודה המושלם בעזרת דרישות המשרה ובעזרת סוכני החיפוש. מהנקודה הזו הרעיון התחיל להתגבש לכדי מערכת אשר משווה בין מאפיינים של אנשים... וקצת יותר.

על מנת להבין את הרעיון המוביל את המערכת אנו צריכים להבין שכל פעולה שמחפש העבודה עושה משפיעה על פרופיל ההתאמה שלו למשרות וההפך,כלומר פרופיל החיפוש של משרה משתנה כאשר משתמש מבצע פעולה על המשרה. ומהצד של המעסיקים, כל פעולה שהם מבצעים על קורות חיים (מתאים, לא מתאים), משפיעה על פרופיל ההתאמה של המשתמש למישרות דומות.
למעשה המערכת מסתכלת על מחפשי העבודה והמישרות בצורה רוחבית ובאופן כזה שכל פעולה חיצונית שנעשית משפרת את אלגוריתם ההתאמה והוא ימשיך וישתפר עם הזמן.

האלגוריתם

בבואנו לתאר את אלגוריתם ההתאמה בין מחפשי עבודה למישרות, עלינו להכיר את החלוקה של היישויות השונות במערכת. המערכת בונה, מתחזקת ושומרת 6 ישויות שונות של מידע אשר ניגזרות ממקורות מידע העומדים לרשותינו כיום ותהיה מסוגלת להיתמודד עם מקורות מידע חדשים ללא שינוי בקוד.

1. **Clean User Profile** - זהו מבנה נתונים אשר ניגזר מ 2 מקורות מידע, עבור כל משתמש הטכנולוגיה "קוראת" את קורות החיים שלו ומבצעת פעולה אוטומטית שניקראת Entity Extraction (ראה הסבר בהמשך), פעולה זו מחלצת מתוך קורות החיים את כל שמות העצם אשר עונים להגדרות של שם חברה, שם טכנולוגיה, שם עיר, שם בית ספר וכו'. מקור המידע השני הוא רשימת סוכני החיפוש שמחפש העבודה מכניס למנוע החיפוש של החברה. המידע הנאגר במבנה נתונים זה הוא **מידע נקי** המתאר את מחפש העבודה **בדיוק** ע"פ המאפיינים שהוא הגדיר את עצמו. המידע עצמו נשמר במבנה מידע של Vector (רשימה של מאפיינים).

2. **Clean Job Profile** - זהו מבנה נתונים אשר ניגזר מ 2 מקורות מידע, עבור כל משתמש הטכנולוגיה "קוראת" את תיאור המשרה כפי שהוכנס ע"י המעסיק ומבצעת פעולה אוטומטית שניקראת Entity Extraction (ראה הסבר בהמשך). מקור המידע השני הוא סוכני החיפוש שהמעסיק משייך את המשרה

אליהם על מנת להקל על מציאתה בידי מחפשי העבודה. המידע הנאגר במבנה נתונים זה הוא מידע נקי המתאר את המשרה בדיוק כפי שהמעסיק הגדיר אותה. המידע עצמו נשמר במבנה מידע של Vector (רשימה של מאפיינים).

3. **Fuzzy User Profile** - זהו מבנה נתונים אשר מצטבר עבור כל מחפש עבודה והוא למעשה מכיל את צברי המאפיינים של כל המשרות שלהן המשתמש שלח קורות חיים (**Clean jobs**). לדוגמא, למשתמש חדש במערכת, מבנה הנתונים הזה יהיה ריק עד לאחר שהוא יגיש קורות חיים למשרה, ברגע שהוא יגיש קורות חיים למשרות, כל מאפייני המשרות יצטברו במבנה נתונים זה. הרעיון המוביל במבנה נתונים זה הוא שהמשתמשים לא תמיד יזכרו לעדכן אותנו במידע חדש ועדכני על ההעדפותיהם ועלינו מוטלת החובה ללמוד את השינויים מתוך הפעולות שלהם. לדוגמא, (הדוגמא מוקצנת מטעמי ההסבר), נניח שמתכנת העלה בעבר קורות חיים וסימן שסוכני החיפוש שלו יהיו בנושא תיכנות, לאחר זמן מה מגיע הקיץ והוא מתחיל להגיש קורות חיים למשרות של מציל. המערכת אמורה להיות מסוגלת ללמוד לבד שהפרופיל של המשתמש השתנה ועליה להציע לו משרות גם מתחום המצילים. וככל שהמשתמש ישלח יותר קורות חיים למשרות מציל, כך היחס הכמותי בין משרות תיכנות ומשרות מציל ישתנה. אלגוריתם זה מבטיח למעשה שהמערכת תמקד את מאמציה בכל כיוון שמחפש העבודה יכתיב ע"י פעולותיו. המידע נשמר בזיכרון כ- Vector (רשימה של מאפיינים), ונישמר כמבנה של טבלה דינאמית בבסיס הנתונים.

4. **Fuzzy Job Profile** - זהו מבנה נתונים אשר מצתבר עבור כל משרה והוא למעשה מכיל את צברי המאפיינים של כל מחפשי העבודה שהגישו קורות חיים למשרה הספציפית (**Clean Users**). לדוגמא, למשרה חדשה במערכת, מבנה נתונים זה יהיה ריק עד לאחר שלפחות מחפש עבודה אחד הגיש קורות חיים למשרה. הרעיון המוביל מאחורי מבנה נתונים זה הוא למפות באופן אוטומטי את הפרופיל הממוצע של כל מחפשי העבודה אשר חשבו שהם מתאימים למשרה זו. המערכת משתמשת במבנה נתונים זה על מנת לחשב את ההיסתברות הסטטיסטית שמחפש העבודה יתקבל למשרה בהינתן היצע קורות החיים שנישלחו למשרה זו. על מנת להבין את הצורך עלינו לדמיין דוגמא(הדוגמא מוקצנת מטעמי הסבר), מעסיק העלה משרה שאחד מהמאפיינים שלה הוא תואר ראשון, ושוק התעסוקה חלש והוא קיבל עד כה 3 קורות חיים של מחפשי עבודה בעלי תואר דוקטור (הרבה מעל הדרישות באותם תנאים), האם מחפש העבודה שמתגאה "רק" בתואר ראשון יהיה בעל סיכוי סביר להתיקבל למשרה ספציפית זו? כמובן שלא, הסיכוי להתיקבל למשרה תלוי בתנאי השוק, מבנה נתונים זה יחשוף לפנינו את יחס ההיצע והביקוש של כל משרה באופן אוטומטי. המידע נשמר בזיכרון כ- Vector (רשימה של מאפיינים), ונישמר כמבנה של טבלה דינאמית בבסיס הנתונים.

5. **Rating Table** - מבנה נתונים חשוב זה מחזיק שלישיות מידע בצורת <UserID, JobID, Rating>, מבנה זה מתעדכן בכל מקרה אשר בו משתמש מבצע אינטראקציה מול משרה, לדוגמא, אם משתמש שלח קורות חיים למשרה תיווצר שורה חדשה עם דירוג 1, ואם המשתמש יסמן את המישרה כמועדפת או ישלח שוב קורות חיים, הדירוג של משרה זו יעלה ל 2. כאן בא לכדי ביטוי מקור מידע חשוב המגיע מכיוון המעסיקים, המעסיקים מצידם, כאשר הם מקבלים קורות חיים, עומדת לפניהם האפשרות לתייק את קורות החיים תחת 3 קטגוריות: 1) המועמד מתאים למשרה, במקרה זה הדירוג של שלישית המידע יעלה ל 3, דבר המסווג אותה כמתאימה מאד. 2) המועמד לא מתאים למשרה, במקרה זה הדירוג של שלישיית המידע יוכפל ב 1-, דבר אשר מסווג אותה כמאד לא מתאימה. 3) המועמד לא נשקל עדיין, במקרה זה הדירוג נישאר כפי שהיה. המידע נישמר במבנה של טבלה דינאמית בבסיס הנתונים.

6. **Related User Table** - מבנה נתונים זה מחזיק מידע על משתמשים דומים כפי שמצתייר מ Rating Table. לדוגמא, אם שני משתמשים שלחו קורות חיים למישרות דומות ויש להם מאפיינים דומים הם

יחשבו מחוברים. לכל משתמש במערכת אנחנו נחבר עוד 5 משתמשים אשר איתם הוא חולק את מספר המאפיינים הרב ביותר. הרעיון המוביל מאחרי הטבלה הזו הוא שעל מנת לקבל כמות ראשונית של מישרות המתאימות למחפש העבודה עלינו לחפש בטבלה זו את 5 מחפשי העבודה הדומים ביותר למשתמש הקיים ולאסוף לרשימה אחת את כל המישרות שאליהם הם הגישו קורות חיים. רשימה זו של משרות תהייה הבסיס שממנו ייגזרו המישרות המוצעות למחפש העבודה הנוכחי. גישה זו מבטיחה שבהינתן שפרופיל המשתמש יתשתנה בעתיד, גם תוצאות החיפוש ישתנו וישקפו את ההעדפות החדשות שלו. המידע נישמר במבנה של טבלה דינאמית בבסיס הנתונים.

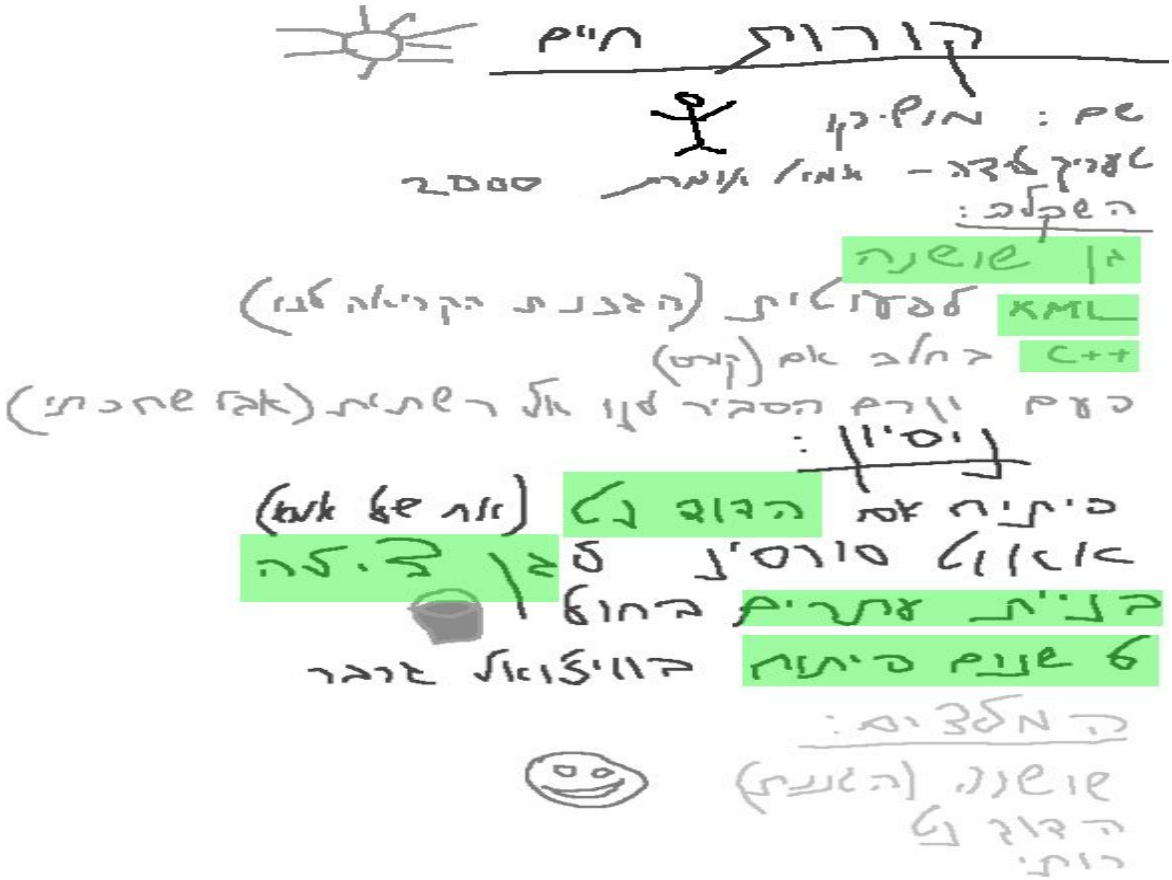
7. **Related Job Table** - מבנה נתונים זה מחזיק מידע על משרות דומות. לדוגמא, אם שתי משרות חולקות אותם מאפיינים ובנוסף שתי המשרות משכו משתמשים דומים אשר שלחו קורות חיים, שתי משרות אילו יחשבו מחוברות.

Entity Extraction Process

הצורך

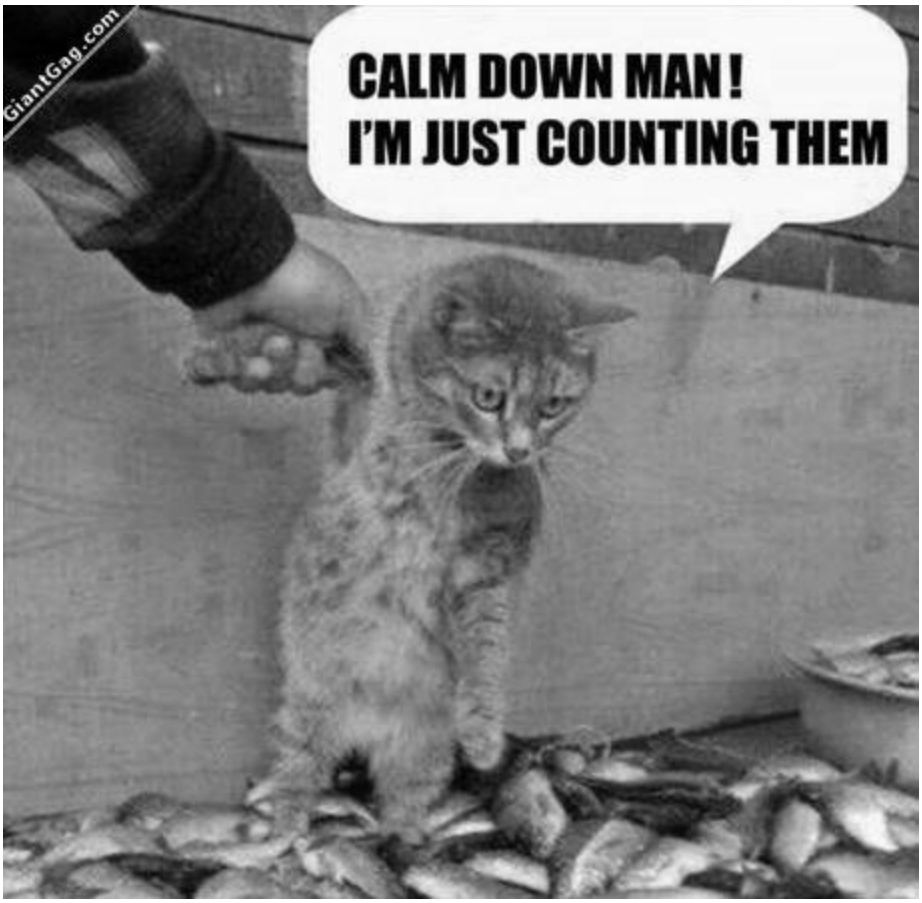
המערכת ניסמכת בכבדות על המידעים המגיעים הן מכיוון מחפשי העבודה והן מכיוון המעסיקים בצורת "סוכני החיפוש", מאפיינים אילו מסומנים ידנית ומאפשרים להציע התאמה קטגורית בין מחפשי עבודה לבין משרות. ועד כמה ששיטה זו מאפשרת חיבורים מדוייקים, היא לעולם לא תוכל להקיף את מגוון המשרות הקיים בעולם האמיתי. המערכת כיום מכילה כ 970 מאפיינים אפשריים, אבל בראותינו את עולם התעסוקה האמיתי אנו מבינים שהוא עולם דינאמי ומשתנה באופן תדיר ועלינו לבנות מנגנון אשר יפצה על חוסר היכולת שלנו להציג מנגנון ידני משתנה בעל אלפי סוכנים אפשריים.

הפתרון



לדוגמא, בעיקבות שינויים בעולם הטכנולוגי נוספה לאחרונה קטגוריית תיכנות חדשה הניקראת Ruby On Rails, זוהי טכנולוגיה חדשנית לכתיבת אפליקציות אינטרנט. לטכנולוגיה זו אין מאפיין חיפוש במערכת שלנו, ומעסיקים המחפשים מועמדים בתחום זה נאלצים להישתמש במאפיין חיפוש כללי של פיתוח לאינטרנט. גם מחפשי עבודה המתמחים בתחום זה אינם יכולים לסווג את עצמם כהלכה, דבר שמתסכל את כל המשתמשים במערכת כיום.
כמובן שהיינו יכולים להוסיף את הקטגוריה הספציפית הזו ל 970 מאפייני החיפוש במערכת, אבל... אם נוסיף את **כל** תתי הקטגוריות האפשריים בשוק התעסוקה נגמור עם ממשק משתמש של מיליוני ערכים, דבר שיעלה את רמת הדיוק, אבל יגביר מאד את רמת התיסכול בהזנת מישרה חדשה או בהזנת קורות חיים.
כלומר, עלינו ליצור מנגנון אשר יהיה מספיק רגיש כדי לזהות מאפיינים חדשים הנובעים מתוך המידע המוגש לנו ב 2 צורות שונות, ע"י מחפשי העבודה בצורת קובץ קורות החיים וע"י המעסיקים בצורת תיאור המשרה. לאחר זיהוי המאפיינים, עלינו לשייך את המאפיינים החדשים למשרה או לקורות חיים שממנו הם נבעו. וכל זאת באופן אוטומטי אשר יהיה שקוף למשתמש.
מנגנון זה יאפשר לנו לקבל את רמת הדיוק של אלפי מאפיינים חדשים ללא הצורך לחשוף את המשתמש לממשק בעל אלפי כפתורים.

השיטה



כדי להשיג את מטרתינו זו עלינו לפנות לשיטות הנהוגות בניתוח שפה טבעית, וספציפית לאלגוריתם מאד פופולרי בקרב מנועי חיפוש, הניקרא **(TF-IDF(Term Frequency inverse Document Frequency**.
אלגוריתם זה מחשב את השכיחות היחסית של מילה במסמך לעומת השכיחות היחסית של המילה בכל המאגר, לדוגמא, אם נספור כמה פעמים מופיעה המילה "את" במסמך בשפה העיברית נטעה לחשוב שהמילה "את" היא מילה מאד חשובה בגלל כמות הפעמים הרבה שהיא מופיעה במסמך. הדרך לאזן את חשיבות המילה היא לספור כמה פעמים היא מופיעה בכל המסמכים, בדוגמא שלנו המילה "את" מופיעה הרבה פעמים בכל המסמכים וכאשר

מחלקים את כמות הפעמים שמילה מופיעה במסמך מול כמות הפעמים בשהמילה מופיעה בכל המסמכים, המילה "את" תקבל חשיבות נמוכה מאד. בשיטה זו אנו ננתח את המילים, צמדי המילים וביטויים קפואים בשפה מכל קורות החיים ומכל המשרות המוצעות על מנת להוציא את המאפיינים החשובים. אלפי המאפיינים המתקבלים יחולצו באופן אוטומטי ויתווספו ל 970 המאפיינים הקיימים במערכת וירחיבו את מרחב החיפוש וההתאמה בין מחפש עבודה למישרות. יתרון בנוי בשיטה זו הוא שהמאפיינים השונים מתקבלים כאשר לכל מאפיין מוצמד המשקל היחסי שלו, מידע זה יאפשר לנו להגיע לרמת התאמה גבוהה מאד בין המאפיינים השונים.

<u>האלגוריתם הכללי</u>

לאחר שהיכרנו את כל המרכיבים במערכת עלינו להבין איך כל הרכיבים מתרכבים לכדי מערכת אחת.

להלן תהליך יצירת פרופיל מאפיינים עבור מחפש עבודה:

1. מחפש עבודה מעלה קורות חיים לאתר - המערכת מנתחת את קורות החיים במנגנון ה Entity Extraction ומייצרת רשימת מאפיינים עם משקלים המותאמת למחפש העבודה.
2. מחפש עבודה בוחר רשימת סוכני חיפוש - המערכת מתייחסת לסוכני החיפוש כמאפיינים נוספים ומכילה עליהם את מנגנון ה TF-IDF על מנת לוודא שהמשקל היחסי הנכון ינתן לכל סוכן. רשימת מאפיינים זו ביחד עם רשימת המאפיינים מסעיף 1 מרכיבה את מבנה הנתונים Clean User Profile.

להלן תהליך יצירת פרופיל מאפיינים עבור משרה:

3. מעסיק מעלה משרה לאתר - המערכת מנתחת את טקסט הדרישות של המשרה במנגנון ה Entity Extraction ומייצרת רשימת מאפיינים עם משקלים המותאמת למשרה.
4. מעסיק בוחר רשימת סוכני חיפוש - המערכת מתייחסת לסוכני החיפוש כמאפיינים נוספים ומכילה עליהם את מנגנון ה TF-IDF על מנת לוודא שהמשקל היחסי הנכון ינתן לכל סוכן. רשימת מאפיינים זו ביחד עם רשימת המאפיינים מסעיף 3 מרכיבה את מבנה הנתונים Clean Job Profile.

להלן תהליך עידכון פרופיל מאפיינים עבור מחפש עבודה:

5. מחפש עבודה שולח קורות חיים למשרה - המערכת מוסיפה ערך לטבלת ה Rating Table (ראה הסבר למנגנון בסעיף הרלוונטי).
6. המערכת "מטביעה" חותם על פרופיל המשתמש בדמות הערכים של המשרה (Clean Job Profile), חותם זה ואחרים מצטברים בפרופיל המשתמש הניקרא Fuzzy User Profile.
7. המערכת מחשבת מחדש ומעדכנת את טבלת Related Users Table ב - 5 המשתמשים הדומים ביותר למשתמש הנוכחי. <userID, Related userID 1, ..2, ..3, ..4, ..5>.

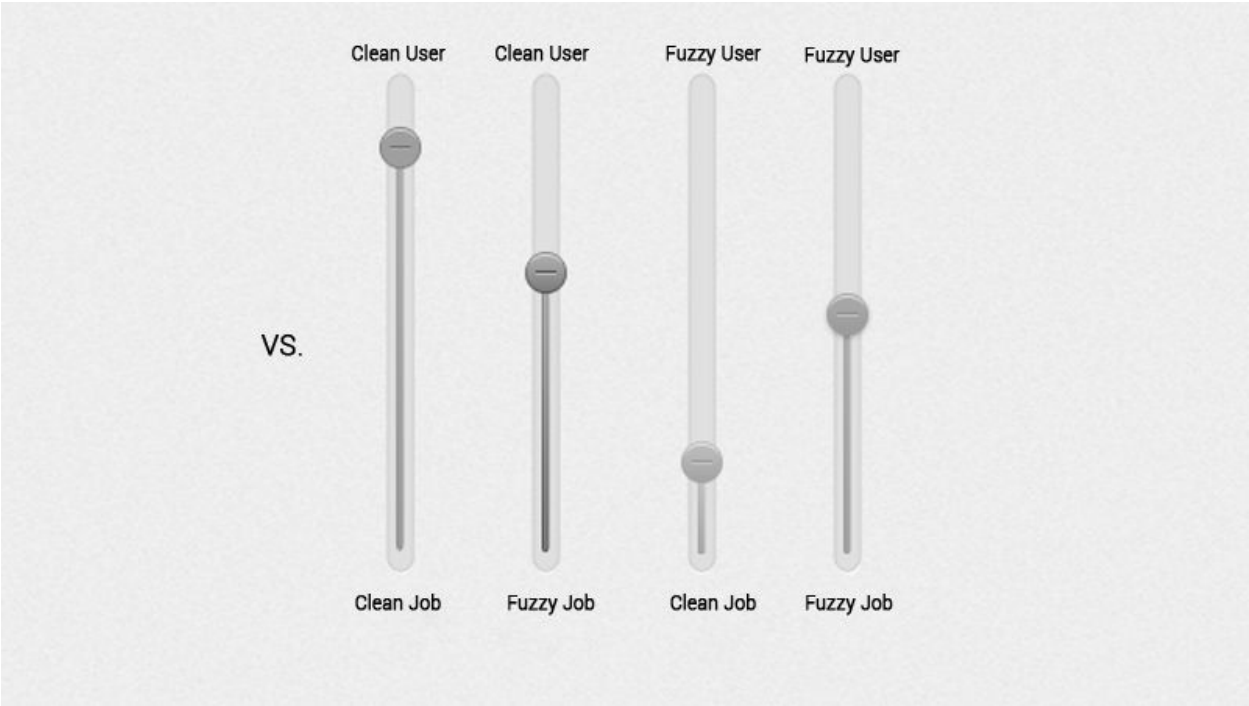
להלן תהליך עידכון פרופיל מאפיינים עבור משרה:

8. מחפש עבודה שולח קורות חיים למשרה - המערכת "מטביעה" חותם על המשרה בדמות הערכים של המשתמש (Clean User Profile), חותם זה ואחרים מצטברים בפרופיל המשרה הניקרא Fuzzy Job Profile.
9. המערכת מחשבת מחדש ומעדכנת את טבלת Related Jobs Table ב - 5 המשרות הדומות ביותר למשרה הנוכחית. <jobID, Related jobID 1, ..2, ..3, ..4, ..5>.

להלן תהליך מציאת מגוון משרות עבור אלגוריתם הדירוג (לא כל המישרות רלוונטיות בשלב זה):

10. מחפש עבודה נכנס לאתר - (אני מניח שקורות החיים וסוכני החיפוש כבר חושבו במערכת), המערכת מביאה מטבלת Related Users את 5 המשתמשים הדומים ביותר למשתמש הנוכחי.
11. עבור כל משתמש דומה שקיבלנו, אנו מביאים את כל המישרות שהוא הגיש להם קורות חיים עד לקבלת מאגר משרות אשר מתוכם ניבחר את המתאימות ביותר להצגה לפני מחפש העבודה.
12. על מנת להעשיר את מגוון המשרות לטובת אלגוריתם הדירוג, עבור כל משרה שהיתקבלה מהסעיף הקודם, אנו מוסיפים למאגר את 5 המישרות הדומות לה ביותר.

<u>אלגוריתם הדירוג</u>



התוצאה של אלגוריתם זה תהיה רשימת משרות להצגה למחפש העבודה, הרשימה תהיה מסודרת בסדר דירוג יורד של רלוונטיות אבל תהיה ניתנת למיון ע"פ פרמטרים שונים.

1. בשלב זה אנחנו מקבלים רשימת משרות מתאימות למחפש העבודה.
2. עבור כל משרה אנו מבצעים את החישובים הבאים:
   a. מה מידת ההתאמה בין Clear User Profile - Clear Job Profile
   מידה זו היא מידת התאמה ב% בין המאפיינים של המשרה למאפיינים של המשתמש.

   b. מה מידת ההתאמה בין Clear User Profile - Fuzzy Job Profile
   מידה זו היא מידת התאמה ב% בין המאפיינים של המשתמש לבין ממוצע המאפיינים של המשתמשים אשר הגישו קורות חיים למשרה זו.

   c. מה מידת ההתאמה בין Fuzzy User Profile - Clear Job Profile
   מידה זו היא מידת התאמה ב% בין ממוצע המאפיינים של המשתמש לבין המאפיינים של משרה זו.

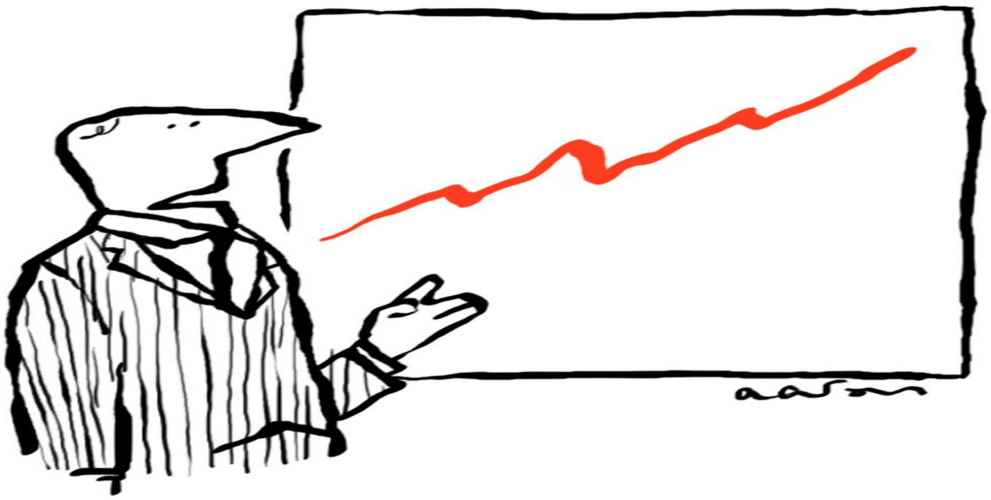
   d. מה מידת ההתאמה בין Fuzzy User Profile - Fuzzy Job Profile
   מידה זו היא מידת התאמה ב% בין ממוצע המאפיינים של המשתמש לבין ממוצע המאפיינים של המשתמשים אשר הגישו קורות חיים למשרה זו.

3. בשלב זה עלינו לקבוע את מידת ה Threshold עבור מחפש העבודה, כלומר, עד כמה מחפש העבודה מוכן להתגמש עבור המישרה הנכונה עבורו. כדי לחשב מידה זו עלינו:
   a. להביא את רשימת המישרות שהמשתמש הגיש להם קורות חיים בעבר.
   b. לחשב את 4 המידות בין המשרות שהמשתמש הגיש להם קורות חיים בעבר ע"פ סעיף 2.
   c. אנחנו נחשב את ממוצע ההתאמה ביו המשרות.

d. ממוצע ההתאמה ישמש כסף תחתון עבור מנגנון הדירוג, כלומר, משרה מוצעת אמורה לקבל מידת התאמה גבוהה יותר מהמידה שהמשתמש בחר בעצמו.
e. התוצאה של סעיף זה תהיה רשימת המשרות בעלות התאמה ע"פ הסטנדרטים שנקבעו ע"פ הצעות עבודה שמחפש העבודה שלח להן קורות חיים בעבר.

4. בשלב זה עלינו למיין ב 4 ממדים את התוצאות ע"פ מידות ההתאמה שחושבו בסעיף 2.
5. התוצאה של סעיף 4 מוחזרת לתצוגה כרשימת משרות ממוינת אשר מויינה ע"פ העדפותיו של מחפש העבודה.

מערכת Data Analytics



"This red line indicates the change in this red line over a period of time."

אני מציע לבנות מערכת אנליטיקס אשר תעקוב ותתעד בזמן אמת את:

1. היחסים בין מועמדים למשרות.
2. חיתוך משתמשים ע"פ המאפיינים שנאספו.
3. חיתוך משתמשים ע"פ המישרות שהם העדיפו.
4. חיתוך משרות ע"פ חתך המאפיינים של המועמדים.

מערכת כזו יכולה לשפר משמעותית את אפשרויות החיפוש המתקדם של משרות או של מועמדים. החברה ע"פ שיקול דעתה תוכל למכור שירותי פרימיום של ייעוץ והכוונה ע"פ המערכת. החברה תוכל גם למכור שירותי גישה לאנליטיקס למעסיקים או למחפשי עבודה רציניים יותר.

מערכת אנליטיקס תאפשר למעסיקים לברור ולזקק את המועמדים לפני קריאת קורות החיים, אפשרות שתגביר את מהירות גיוס עובדים חדשים. מאפיינים נוספים יוספו ע"פ הצורך.

ארכיטקטורה כללית + עידכון אוטומטי

המערכת תהיה כתובה בטכנולוגיית NET. של מיקרוסופט וספציפית ב #C. הסיבה העיקרית לבחירה בטכנולוגיה זו היא האוריאנתציה הכללית של הטכנולוגיה בחברה כיום. סיבה נוספת לבחירה בטכנולוגיה זו היא החיבור הנקי והאלגנטי של טכנולוגיית NET. ומוצרי מייקרוסופט מתקדמים כ Azure Machine Learning Platform.

החיבור למערכת הקיימת בחברה

מתוך ההבנה שמערכת זו (לפחות בהתחלה) תשמש כמערכת תומכת במערכת ההתאמה הקיימת בחברה, ומתוך זהירות מחשש לפגום בחוויה הכללית של המשתמשים בזמן ההטמעה של הטכנולוגיה החדשה, תכננתי את המערכת כך שאין לה ממשק חיבור עם המערכת הקיימת בחברה. מעבר לזה, כיום קיימת בחברה מערכת חוקי BI שאני לא מעוניין להשפיע עליה או לשכפל אותם, לדוגמא, איך ומתי משרה יורדת מהאוויר? עד מתי מחפש העבודה נחשב כמשתמש פעיל? וכו.
המערכת תעבוד בשיטה של עידכונים אוטומטיים בכל X דקות\שעות, לדוגמא, בכל שעה המערכת תשאב נתונים מבסיס נתונים יעודי המכיל את הטבלאות הבאות:

1. UserJobs - על מנת להרכיב את Rating Table.
2. UserAgents - על מנת להתעדכן במאפייני חיפוש חדשים של מחפש העבודה.
3. Resumes - על מנת להתעדכן בשינוי קורות החיים של מחפש העבודה.
4. Jobs - על מנת להתעדכן בשינויים במישרות.
5. JobAgents - על מנת להתעדכן במאפייני חיפוש חדשים של המשרה.

אני מצפה לקבל טבלאות **מעודכנות** רק במשרות רלוונטיות ורק במשתמשים פעילים.
מאחר שהמערכת היא עתירת חישובים, המערכת מתוכננת כך שכל החישובים יעשו ב IDLE TIME, עבור כל מחפש עבודה פעיל המערכת תכין מראש רשימת משרות מומלצות, וברגע שהמשתמש יכנס למערכת כל החישובים כבר חושבו ולא יחושבו על חשבון זמנו של המשתמש.
בעת עידכון, המערכת תחשב את כל ההמלצות מחדש בהתאם למידת עידכון המידע הבסיסי, כלומר כאשר מתבצעת פעולות ישירות בין מחפשי עבודה למשרות.
התוצאה של המערכת היא טבלת Job Recommendation בפורמט הבא, <,...
.<UserID,RecommendedJobID1,..2..3

The Flow

1. עבור כל מחפש עבודה בטבלת Resumes (אני מצפה לקבל משתמשים פעילים)
   a. האם משתמש חדש - מחפשים בטבלת ה Caching Users (טבלת עזר פנימית)האם שמור Vector עבור המשתמש:
      i. לא קיים - כלומר המשתמש הוא חדש:
         1. עלינו לשלוח את שדה AllText (קורות חיים)מטבלת Resumes לרכיב ה Entity Extraction להוצאת Vector של המאפיינים מקורות החיים. בהינתן ששדה זה ריק, לא עושים פעולה.
         2. עלינו להפוך את שדה ה CTR של מחפש העבודה ל Vector.

3. עלינו לעדכן את טבלת Caching Users ב 2 ה Vectors .

ii. קיים - משתמש קיים במערכת

1. עלינו לבדוק האם קורות החיים או ה CTR השתנו, בהינתן שהישתנו עלינו לשלוח להוצאת Vector מאפיינים חדשים, ולעדכן את טבלת Caching Users במאפיינים החדשים.

2. עבור כל משרה בטבלת Jobs (אני מצפה לקבל משרות פעילות בלבד)

a. האם משרה חדשה - מחפשים בטבלת ה Caching Jobs (טבלת עזר פנימית)האם שמור Vector עבור המשרה:

i. לא קיים - כלומר המשרה היא חדשה:

1. עלינו לשלוח את שדה job requirements מטבלת Jobs לרכיב ה Entity Extraction להוצאת Vector של המאפיינים מטקסט המשרה.
2. עלינו להפוך את שדה ה CTR של המשרה ל Vector.
3. עלינו לעדכן את טבלת Caching Jobs ב 2 ה Vectors .

ii. קיים - משרה קיימת במערכת

1. עלינו לבדוק האם טקסט המשרה או ה CTR השתנו, בהינתן שהישתנו עלינו לשלוח לרכיב ה Entity Extraction להוצאת Vector מאפיינים חדשים, ולעדכן את טבלת Caching Jobs במאפיינים החדשים.

3. מייצרים את טבלת הרייטינג, עבור כל רשומה ב UserJobs :

a. האם המעסיק סימן את המועמד כ "מתאים", בהינתן שכן אנחנו מעלים ב +1 את הרייטינג של הרשומה.

b. האם המעסיק סימן את המועמד כ "לא מתאים", בהינתן שכן אנחנו מכפילים ב 1- את הרייטינג הקיים.

4. מייצרים את טבלת UsersFeatures, JobFeatures מvector המאפיינים של Caching Users ו Caching Jobs בהתאמה.

5. שולחים את 3 הטבלאות שייצרנו ל AZURE ML על מנת לייצר את טבלת RelatedUsers ו RelatedJobs בהתאמה.

6. על מנת לייצר רשימת הצעות עבודה עבור כל משתמש ממשיכים ע"פ <u>סעיף האלגוריתם (למעלה)</u>.

7. מייצרים את טבלת InActiveUsers - משתמשים שאינם פעילים, מאפיין זה זה נדרש על מנת להוריד את המשתמשים האילו מטבלת Related Users. כדי לייצר טבלה זו אנחנו נשווה בין טבלת Caching Users הקיימת לבין טבלת Resumes שהיתקבלה, בהינתן שמשתמש קיים ב Caching Users ולא קיים ב Resumes, אנחנו מסיקים שהוא לא פעיל ומסמנים אותו כלא פעיל. (לא מוחקים!!!)

8. מייצרים את טבלת InActiveJobs - משרות שאינן פעילות, מאפיין זה נדרש על מנת להוריד את המשרות האילו מטבלת המישרות המוצעות ואולי להציע משרות חדשות למחפשי העבודה במקום המשרה שירדה. כדי לייצר טבלה זו אנחנו נשווה בין טבלת Caching Jobs הקיימת לבין טבלת Jobs שהיתקבלה, בהינתן שמשרה קיימת ב Caching Jobs ולא קיימת ב Jobs, אנחנו מסיקים שהמשרה לא פעילה ומסמנים אותה כלא פעילה. (לא מוחקים!!!)

System Architecture
The Processes

MainProcess
Public Class

- MainProcess()
- Main(resumesTable: List<Resumes>, jobsTable: List<Jobs>, userJobsTable: List<UserJobs>) : List<List<JobRecommendation>>

EntityExtractionProcess
Public Class

- EntityExtractionProcess()
- DoEntityExtractionProcess(allUsersInSystem: List<User>, allJobsInSystem: List<Job>) : Void
- AssignNewWeights(cleanedAllDocumentsInSystem: List<TextDocument>) : Void
- ExtractAllDocuments(allUsersInSystem: List<User>, allJobsInSystem: List<Job>) : List<TextDocument>

JobRecommendationProcess
Public Class

- maxRecommendationsForUser: Int32

- JobRecommendationProcess()
- JobRecommendationProcess()
- GetRecommendations(ratingTable: List<RatingTable>, relatedUsers: List<RelatedUsers>, relatedJobs: List<RelatedJobs>, allUsers: List<User>, allJobs: List<Job>) : List<List<JobRecommendation>>
- OrderInDescendingOrderBySimilarities(jobRecommendations: List<JobRecommendation>) : Void
- ComputeSimilarities(jobRecommendation: JobRecommendation, job: Job, user: User) : Void
- CalculateUserThreshold(userJobs: List<Job>) : Double
- GetSimilarityScore(first: List<Vector>, second: List<Vector>) : Double
- getJobsForUser(userID: Int32, ratingTable: List<RatingTable>, allJobs: List<Job>) : List<Job>

RatingTableCreator
Public Class

- RatingTableCreator()
- CreateRatingTable(UserJobsTable: List<UserJobs>) : List<RatingTable>

RelatedUsersJobsCreator
Public Class

- RelatedUsersJobsCreator()
- CreateRelatedUsersTable(users: List<User>, ratingTable: List<RatingTable>) : List<RelatedUsers>
- CreateRelatedJobsTable(jobs: List<Job>, ratingTable: List<RatingTable>) : List<RelatedJobs>

Data Components

UserJobs
Public Class

- UserID: Int32
- JobID: Int32
- CvID: Int32
- TabIndex: Int32

- UserJobs()

Jobs
Public Class

- JobID: Int32
- text: String
- CTR: String

- Jobs()

Resumes
Public Class

- UserID: Int32
- text: String
- CTR: String

- Resumes()

RatingTable
Public Class

- UserID: Int32
- JobID: Int32
- Rating: Int32

- RatingTable()

RelatedUsers
Public Class

- UserID: Int32
- mRelatedUsers: List<Int32>
- SimilarityScores: List<Double>

- RelatedUsers()

RelatedJobs
Public Class

- UserID: Int32
- mRelatedJobs: List<Int32>
- SimilarityScores: List<Double>

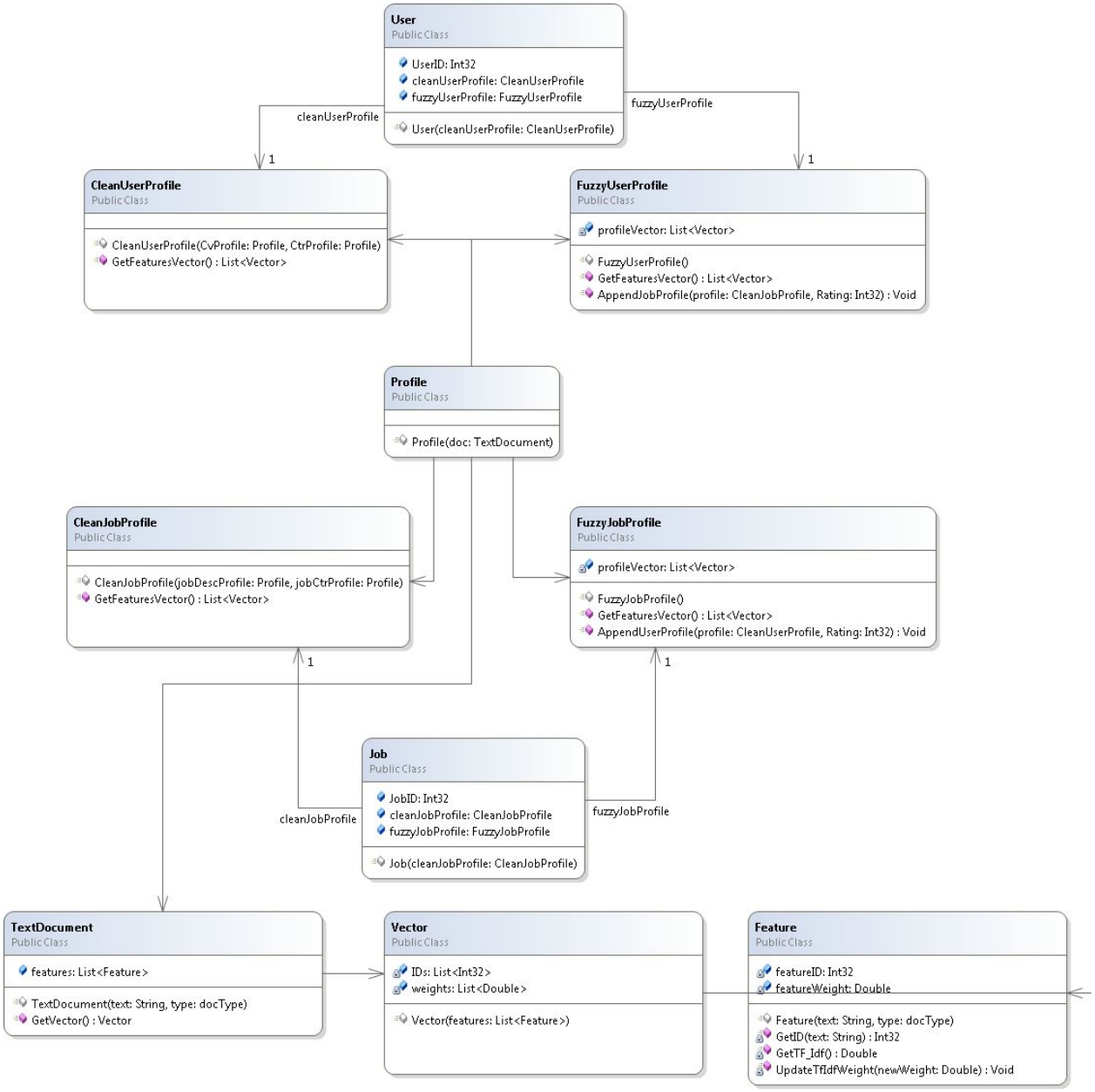
- RelatedJobs()

JobRecommendation
Public Class

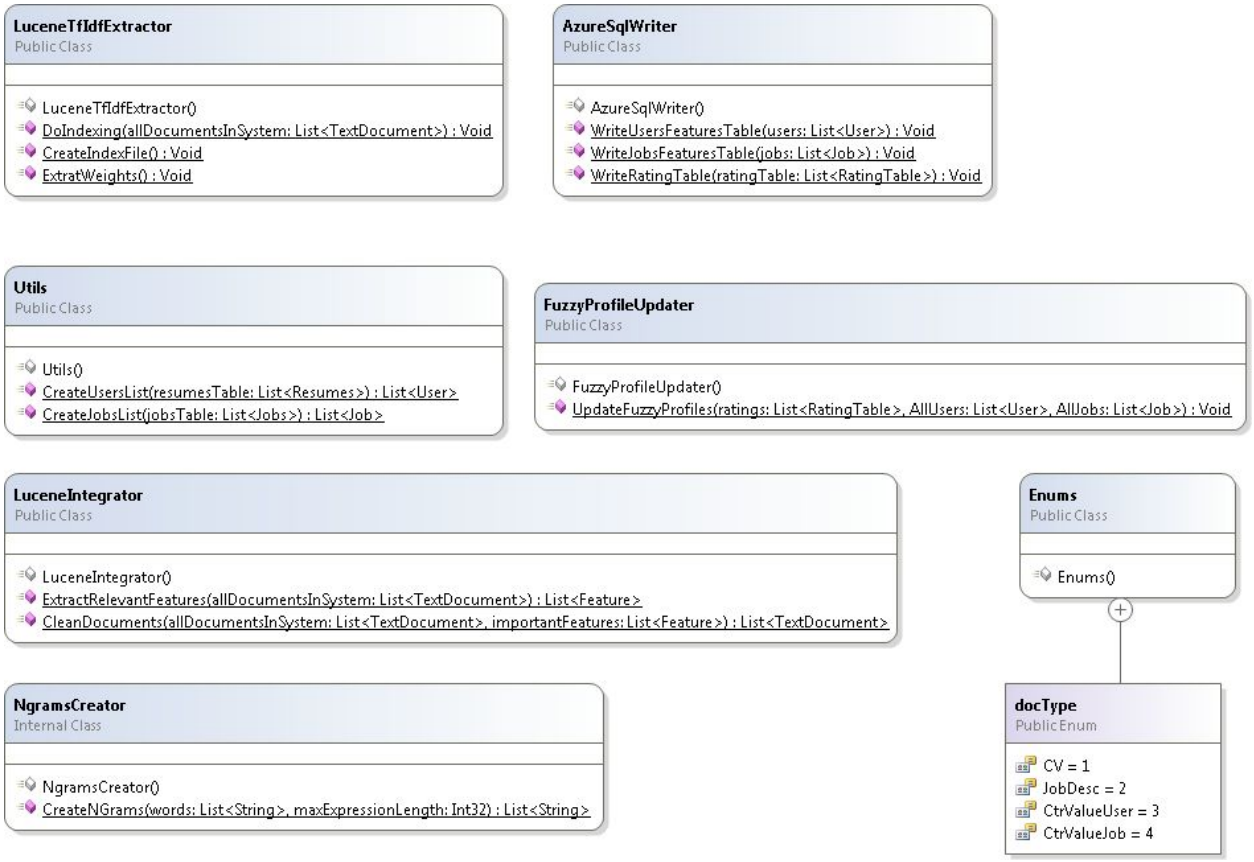
- UserID: Int32
- JobID: Int32
- CleanUserProfile_CleanJobProfile_similarity: Double
- CleanUserProfile_FuzzyJobProfile_similarity: Double
- CleanJobProfile_FuzzyUserProfile_similarity: Double
- FuzzyJobProfile_FuzzyUserProfile_similarity: Double

- JobRecommendation(UserID: Int32, JobID: Int32)

The Entities



Utility Components



Azure Machine Learning Platform as framework

על מנת לבצע חלק מהחישובים המורכבים אנו נשתמש בטכנולוגיית Azure Machine Learning, מייקרוסופט הציגה כלי מדהים אשר מקצר משמעותית את זמן הפיתוח של מערכות מורכבות מתמטית. בכלי זה אנו נוכל לאמן את המערכת ללמוד את המאפיינים של כל משתמשי העבר ולהציע ניבוי הסתברותי עבור משתמשים חדשים.
בטכנולוגיה זו אנו יכולים גם לבדוק את אמינות החישובים כאשר ננסה לנבא את תוצאות העבר ולקבל מידע מדוייק לגבי אמינות החישובים במערכת.
בפועל אנו נשתמש בפלטפורמה זו על מנת לבצע את החישובים הבאים:
אנו נשתמש באלגוריתם Score Matchbox Recommender על טבלת Rating Table כדי לחלץ את 2 הטבלאות הבאות:

1. Related Users Table
2. Related Jobs Table

יתרונות למזיני משרות

המערכת תחשוף פונקציה פומבית אשר תשדרג את חווית הזנת המישרה. הפונקציה שתיקרא GetFeatures תקבל כארגומנט "תיאור משרה טקסטואלי" ותחזיר רשימת מאפייני CTR ממוינת עם משקלים עבור כל מאפיין. אנשי ה UI בחברה יוכלו להציע למזיני המישרות באתר להשתמש במאפיינים שנוצרו באופן אוטומטי מהמערכת. מזיני המשרות יקבלו את האפשרות לערוך את המאפיינים שהיתקבלו, כלומר להוריד מאפיינים או להוסיף לרשימה מתוך רשימה קיימת.
תהליך זה יאפשר למערכת "ללמוד" אילו מאפיינים חשובים יותר ולהעלות להם את המשקל היחסי, או אילו פחות חשובים ולהוריד להם את המשקל היחסי. הרעיון פה הוא שמזיני המישרות ישמשו כמערכת Evaluation עבור פונקציית GetFeatures

יתרונות למחפש העבודה

היתרונות הבולטים למחפש העבודה הם:

1. המערכת מחזירה הצעות עבודה מאד רלוונטיות בסדר דירוג יורד. הסיבה לזה היא היכולת של המערכת להיסתכל בצורה רוחבית על כל הצעות העבודה ועל כל המועמדים האחרים שהגישו קורות חיים על מנת לשפר לא רק את ההתאמה בין הטקסטים של משרות לקורות חיים אלא **לשפר את ההסתברות** שמחפש העבודה באמת יתקבל למשרה שהוא רוצה ע"י הסתכלות על חתך המאפיינים של המגישים קורות חיים למשרה מסויימת.
2. המערכת מתאימה את עצמה אישית עבור כל משתמש, כלומר, משתמש אינו חייב לסמן סוכני חיפוש חדשים, אלא המערכת תהייה מספיק רגישה כדי לזהות שינויים בדפוס המשרות שמחפש העבודה חיפש וכמובן להכיל את השינויים האלה על הצעות העבודה החדשות.
3. המערכת תוכל בעתיד להציע הצעות להסבת מקצוע, אנו נשיג זאת ע"י השוואת המאפיינים של המשתמש למאפיינים של משתמשים אשר ביצעו הסבת מיקצוע דומה בעבר.
4. המערכת אינה מוחקת נתונים והיא תלווה את מחפש העבודה לאורך הקריירה שלו.

יתרונות למעסיק

היתרונות הבולטים למעסיק הם:

1. הדיוק הרב של המערכת בהתאמת מחפשי עבודה למשרות הוא יתרון רב עבור המעסיקים מאחר והם יקבלו מועמדים מתאימים יותר לדרישותיהם ולא יאלצו לקרוא קורות חיים מיותרים.
2. מנגנון הזנת סוכני החיפוש הידני עבור משרות ישודרג משמעותית מאחר והשימוש ב Entity Extraction יאפשר למערכת להציע לעורך המשרה לשלב מאפיינים אוטומטים שחולצו מתוך טקסט תיאור המשרה.
3. מנגנון המאפיינים האוטומטי יאפשר לפרסם מישרות לקהלים ממוקדים יותר.

4. מאחר שהמערכת שומרת על מידעים מהעבר ולכל משרה יש משרות דומות, מערכת תוכל בעתיד להציע לעורך משרה לראות חתך של מחפשי עבודה שפנו למשרה דומה בעבר. מנגנון זה יאפשר לעורך המשרה לזקק את תיאור המשרה כך שרק מועמדים מסוג מסויים ייפנו למשרה.
5. המערכת תאפשר בעתיד לייצר Analytics dashboard עבור מעסיקים כדי להעריך את המאפיינים של מחפשי העבודה ולסנו ולזקק רשימות משתמשים.

יתרונות לחברת All-Jobs

היתרונות הבולטים לחברה הם:

1. מחפשי עבודה מרוצים בדמות הצעות רלוונטיות למשרות.
2. מעסיקים מרוצים בדמות זמן גיוס עובדים מקוצר.
3. מזיני משרות מרוצים בדמות חילוץ מאפייני חיפוש באופן אוטומטי מטקסט המישרה.
4. מעסיקים מרוצים אשר יוכלו להשתמש במערכת האנליטיקס על מנת לזקק את דרישות המשרה בהתאם למאגר קורות החיים בזמן אמת.
5. מעסיקים מרוצים אשר יוכלו להשתמש במערכת האנליטיקס על מנת לחתוך ולקטלג את המועמדים אשר פנו אליהם וזאת לפני קריאת קורות החיים.
6. מעבר לתהליכי טקסט פשוטים, המערכת תומכת בכל השפות, דבר שיאפשר לחברה לחקור אפשרויות התרחבות לשפות נוספות.
7. המערכת לומדת באופן אוטומטי מאפיינים חדשים, כלומר אין צורך לעדכן את הקוד במיקרה של הוספת מאפיינים.
8. המערכת תדע לבדוק את עצמה בדמות Sanity Checks אשר יזהו שינויים משמעותיים בזרם המידע הנכנס וישלחו התראות בדמות מייל ל system administrator.
9. המערכת בנויה להישתפר עם הזמן והשימוש.
10. לדעתי (כמובן שצריך להתייעץ עם עורך פטנטים) המערכת מספיק חדשנית כדי להגיש פטנט, פעולה שתגדיל משמעותית את איכות הנכסים האיטלקטואלים של החברה.

מטרות הפרוייקט

בסעיף זה אתייחס למטרות הפרוייקט כפי שהוגדרו ואיך החלקים השונים במסמך זה עונים על מטרות הפרוייקט.

1. **מטרה** - בניית טכנולוגיה אשר תנתח מסמכים (קורות חיים ומשרות) ע"י שימוש בשיטות סטטיסטיות ותחביריות. תוצאת הניתוח תהייה חילוץ מכלול המאפיינים של קורות חיים או משרות.
   מימוש - ראה בבקשה סעיף Entity Extraction Process.

2. **מטרה** - בניית טכנולוגיה אשר תשווה בין מכלול מאפייני קורות חיים למאפייני משרה. תוצאת ההשוואה תהיה מידת ההתאמה האוטומטית בין קורות חיים למשרה.
   מימוש - ראה בבקשה את הסעיף האחרון בסעיף האלגוריתם הכללי.

3. **מטרה** - בניית טכנולוגיה אשר תתאים בין מכלול המאפיינים לקטגוריות הקיימות בחברה כיום.
   מימוש - סעיף זה מכוון לטובת שידרוג מערכת הזנת המישרות, ראה סעיף יתרונות למזיני משרות.

4. **מטרה** - בניית טכנולוגיה אשר תמפה את הקשרים הסטטיסטיים שמתקיימים בין המאפיינים השונים לטובת הרחבה או סינון התאמות בין קורות חיים למשרה. מיפוי זה יאפשר מנגנונים שונים בתצוגת המשתמש (UI).
מימוש - ראה סעיף מערכת Data Analytics.

נסיונות שערכתי

על מנת לבדוק את הנחות היסוד שלי בנוגע לאלגוריתם, ועל מנת לתת לחברה "טעימה" ממה שמצפה לנו, בניתי אב טיפוס למערכת בכתובת הבאה: http://aj.meos1000.com
המערכת מייצרת הצעות עבודה עבור 113,823 משתמשים מתוך מאגר של 25,932 משרות.

אב טיפוס - הסבר:
בדף הראשי אין שום דבר מעניין מעבר לרשימה של משתמשים, לחיצה על משתמש למעשה תפתח עמוד אשר מכיל את הצעות העבודה שהמערכת תציע למשתמש, העמוד מכיל:

1. קורות החיים של המשתמש כפי שהוזנו ע"י המשתמש
2. Recommended Jobs - רשימת מישרות מוצעות למשתמש זה ע"פ המערכת. קו אדום באמצע רשימה זו הוא סימן שציון כל המשרות מתחתיו היה מתחת למידת ה Threshold והם אינן יוצגו למשתמש.
3. User Selected Jobs - רשימת המישרות שהמשתמש בחר ידנית לשלוח להם קורות חיים. קו אדום באמצע רשימה זו הוא סימן שציון כל המשרות מתחתיו היה מתחת למידת ה Threshold והמערכת אינה השתמשה בהם כדי ללמוד על המשתמש.
4. שורות הצבועות בסגול הם הצעות עבודה שהמערכת מציעה למשתמש מתוך המאגר של 25,932 משרות מבלי לדעת שהוא עצמו בחר בהם!!!

הערכת זמן פיתוח

אני מעריך שבהסתמך על:

1. Azure machine learning
2. מערכת Lucene search Open-source
3. כתיבת קוד יעודי למערכת

זמן הפיתוח עבור הרכיבים שפירטתי בהצעה זו הוא X חודשי עבודה.

חלומות

אני מאמין בכל ליבי שעל חברה מובילת דרך ודעה כמו אולג'ובס מוטלת החובה המוסרית לנסות ולשפר את חיי המשתמשים הבוחרים בשירותים שלה.
אני רוצה להציע לקברניטי החברה לשקול בחיוב הוספת:

1. על כל משרה להוסיף שדה חובה של טווח שכר עליון וטווח שכר תחתון.
2. כל מחפש עבודה יחוייב להכניס למערכת טווח שכר עליון וטווח שכר תחתון.

מספרים אלה אינם יחשפו למחפשי עבודה או למעסיקים אלא ישמשו את המערכת כחלק חשוב ומרכזי בנסיון לייצר חיבור בעל סיכוי יותר טוב בין ציפיות השכר של מחפש העבודה מול התקציב הקיים של המעסיק.

אני מאמין שאם יתאפשר למערכת להשתמש במספרי השכר כחלק מתהליך הלמידה, המערכת תוכל לדחוף כלפני המשתמש את המישרות המתאימות ביותר עבורו וגם את המישרות שהשכר בהם הוא גבוה יותר. התהליך הזה למעשה יבעבע את המישרות הטובות ביותר כלפי המשתמשים וכמובן שככל שהרייטינג של משרה יעלה כך היא תקבל יותר מועמדים. מזיני המשרות יקבלו מהמערכת הצעות אוטומטיות לטווחי שכר ממוצעים ע"פ סוג המשרה והדרישות.

תהליך זה יגרום למעסיקים להציע שכר גבוה יותר כדי לגייס מועמדים איכותיים יותר, פעולה שתגרום לעליית השכר הממוצע ברוב התחומים. גם המעסיקים יהנו מהתוספת החדשה בדמות מועמדים אשר ציפיות השכר שלהם מתאימה לתקציב המישרה.
המידע שיאסף בבסיס הנתונים של החברה יהווה את הבסיס לטבלאות השכר הממוצעות המעודכנות במשק.

תודה

מסמך זה נכתב בעזרתו והכוונתו של בוני גרוסמן.
אני רוצה להודות לבוני גרוסמן וליגאל חדד אשר תרמו מזמנם וכישרונם לטובת המערכת.