

אוניברסיטת חיפה

תכנית "מדעי הרוח הדיגיטליים"

במסגרת הקורס: "טקסט בעידן הדיגיטלי"

משימת XML-TEI Corpus

מגיש: עידו רום

ת.ז.: 312143514

לידי המרצה: ד"ר סיני רוסינק

**תוכן עניינים**

[1 מטרת העבודה 3](#_Toc1817608)

[2 XML-TEI 3](#_Toc1817609)

[3 דרך הפעולה שנבחרה 3](#_Toc1817610)

[4 פתרון 3](#_Toc1817611)

[5 אופן פעולת התוכנית 4](#_Toc1817612)

[6 התבניות בהן נעשה שימוש בקוד 4](#_Toc1817613)

[7 בעיות בהן נתקלתי במהלך העבודה ופתרונן 5](#_Toc1817614)

[8 דוגמה לתוצאה 6](#_Toc1817615)

[9 הפונקציה העיקרית בתכנית 7](#_Toc1817616)

# מטרת העבודה

מטרת העבודה היא יצירת קובצי XML-TEI Corpus, על ידי ניתוח קובצי טקסט המאגדים אוסף מכתבים מאת שמריהו לוין. בקובץ ה XML-TEI, תהיה חלוקה לפי מכתבים, ולכל אחד מהם ישמר Metadata רלוונטי.

קובץ המקור מכיל אוסף של מכתבים, ללא חלוקה מסודרת ביניהם – אין סימן ברור לסיום של מכתב ותחילתו של המכתב הבא, וה- Metadata שנדרש למצוא, מופיע בפורמט וסדר שונה במכתבים.

# XML-TEI

XML Text Encoding Initiative - הינו פורמט לייצוג טקסט ושמירת Metadata ע"י שימוש ב XML מורחב. הפורמט פותח לשימוש אקדמי במדעי הרוח ומשמש לשמירת טקסט ו Metadata עבור סוגים רבים של מידע הומני, כמו מכתבים, מחזות, ספרים, מאמרים וכדומה. הפורמט שם דגש על ייצוג של הטקסט ו Metadata אודותיו ולא על אופן הצגת הטקסט, כמו למשל פורמט ה HTML.

XML-TEI, כמו XML, משתמש בתגיות, אך ב- XML-TEI מוגדרות תגיות ספציפיות ומבנה ייחודי לכל סוג של מידע הומני.

# דרך הפעולה שנבחרה

דרך הפעולה שנבחרה היא יצירת תוכנית להמרה של קובץ טקסט המכיל מספר מכתבים ברצף, לקובץ XML, תוך כדי חלוקה למכתבים וזיהוי של Metadata בתוך כל מכתב.

המידע שנדרש למצוא בכל מכתב, כדי לחלק את הטקסט הרציף למכתבים בודדים ולהוציא מכל מכתב את ה- Metadata הוא:

* הנמען
* כתובת הנמען (עיר)
* תאריך המשלוח
* המקום ממנו נשלח המכתב
* חתימת השולח

# פתרון

בחרתי לבצע את המטלה על ידי כתיבת תכנית ב C#. הסיבות לבחירה זו הן:

* אוטומציה של התהליך (לא פעולה ידנית מייגעת)
* לימוד של נושא ניתוח ממוחשב של טקסט
* אימון ב #C

ניתוח ממוחשב של טקסט הוא תחום נרחב, בעבודה זו נגעתי רק בחלקים קטנים מתוכו. החלק הראשון הוא חלוקה של הטקסט לחלקים נפרדים על פי מאפיין מסוים. במקרה שלנו, לאחר בחינה של קבצי המקור, מצאתי שאת  החלוקה למכתבים ניתן לעשות על פי שורת הנמען תוך העזרות בשורה של החתימה לזיהוי של תחילת המכתב הבא.

החלק השני הוא זיהוי של מאפיינים מסוימים בטקסט, כמו חתימה, נמען, תאריך משלוח. הזיהוי בפתרון שלי הוא על פי תבנית של מילים ו / או מספרים.

זיהוי התבניות נעשה על ידי Regular Expression. הבחירה בדרך זו נעשתה מהטעמים הבאים:

* גמישות יחסית של Regular Expression, מאפשר זיהוי של תבניות טקסט מורכבות.
* פישוט הפתרון
* היכרותי עם מנגנון ה Regular Expression

זיהוי התבניות לכל מאפיין נעשה בשני שלבים:

מעבר ידני על הטקסט ואפיון של התבניות לכל חלק, כגון נמען, תאריך משלוח וחתימה.

הפעלת התוכנית ומציאת מקרים בהם התבנית לא מצאה את המאפיין כנדרש ושיפור התבנית.

למשל התבנית הרווחת של שורת הנמען היא: "אל [שם] , [עיר]"

לדוגמה "אל אחד העם, לונדון"

אבל יש גם דוגמאות אחרות, למשל "לאסיפה השנתית של ציוני אמריקה".

בכדי להתגבר על השונות בטקסט אותו מחפשים, תבניות ה Regular Expression, הן מורכבות, ומכילות תת תבניות חלופיות. בתוך תבניות השתמשתי ב Named Groups בכדי לזהות את ה Metadata.

# אופן פעולת התוכנית

התוכנית מכילה לולאה הקוראת את הטקסט, שורה אחת בכל פעם. כל שורה נבדקת מול התבניות למציאת התאמה למאפיין מסוים. במידה ונמצאה התאמה לתבנית הנמען, נרשם ה XML של המכתב הקודם, כולל המאפיינים שלו, והלולאה ממשיכה.

# **התבניות בהן נעשה שימוש בקוד**

נמען והכתובת אליה נשלח המכתב:

^((אל )|(ל))(?<RecipientName>((([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))(  
+(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))\*(,(  
+(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))+)??))(,(?<RecipientLocation>( +(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))+))?\.?$

תאריך ומקום ממנו נשלח המכתב:

^ \*((?<SentFrom>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))(  
+(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))\*) \*,?( \*(?<Date>(\d\d?(\-|–))?(?<day>\d\d?),? (?<month>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))) ((?<year>\d+)))))|((?<Date>(?<day>\d\d?) (?<month>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))  
(?<year>\d+))) \*, \*(?<SentFrom>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))(  
+(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))\*)|((?<Date>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))  
+[א-ת][א-ת][א-ת]"[א-ת]) \*(, \*(?<SentFrom>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))))?\.\*)|((?<SentFrom>(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.))(  
+(([א-ת][\-'’"“”א-ת]\*[’'א-ת])|([א-ת]\.)))\*) \*,? \*(?<Date>(?<day>\d+)\.(?<month>\d+)\.(?<year>\d+))) \*$

חתימה:

((?<From>(שמריה הלוי)(( לעווין)?|( לוין)?)( לעווין)?)|(?<From>(ש\. +)?(לוין))) \*$

# בעיות בהן נתקלתי במהלך העבודה ופתרונן

תהליך בניית התבניות היה הדרגתי, התחלתי עם התבניות הרווחות ביותר בטקסט, הרצתי את התוכנית מול  הטקסט הנתון, ומצאתי שהיא לא מזהה את כל הצרופים. בכל פעם שנתקלתי באפשרות חדשה, הוספתי אותה לתבנית והפעלתי את התוכנית מחדש. ריבוי האפשריות לכל תבנית גרם לכך ש ה Regular Expression לתבניות הוא מורכב.

שימוש בתווים לא טנדרטיים בטקסט, כגון מרכאות מעוגלות (“). הפתרון, הוספת התווים המיוחדים לתבניות.

# דוגמה לתוצאה



# הפונקציה העיקרית בתכנית

private static void ReadSourceFile(string srcFileName, string DstFileName)

{

Match m;

string LetterText = "";

string RecipientName = "";

string RecipientLocation = "";

string SentFrom = "";

string DateSent = "";

string SenderName = "";

string SentDateStr = "";

int NumOfLetters = 0;

string OutputXml = XMLTop();

SenderName = " ";

// open the source file for reading

using (StreamReader streamReader = File.OpenText(srcFileName))

{

string Line;

// loop through each line in the source file

while ((Line = streamReader.ReadLine()) != null)

{

string s = Line;

Line = Line.Replace("#", "").Replace("$", "");

// check for "To" line

m = ToRX.Match(Line);

if (m.Success && !string.IsNullOrEmpty(SenderName))

{

// check if new letter starts in this line

if (!string.IsNullOrEmpty(LetterText))

{

NumOfLetters++;

OutputXml += OneLetterXML(NumOfLetters, SenderName, SentFrom, DateSent, RecipientName, RecipientLocation, LetterText, SentDateStr);

// reset metadata for next letter

LetterText = "";

RecipientName = "";

RecipientLocation = "";

SentFrom = "";

DateSent = "";

SenderName = "";

SentDateStr = "";

}

RecipientName = m.Groups["RecipientName"].Value.Trim();

RecipientLocation = m.Groups["RecipientLocation"].Value.Trim();

LetterText = Line;

}

else LetterText += "\r\n " + Line;

// Check for "Date" line

m = DateRX.Match(Line);

if (m.Success && string.IsNullOrEmpty(SentFrom))

{

SentFrom = m.Groups["SentFrom"].Value.Trim();

DateSent = m.Groups["Date"].Value.Trim();

string ds = m.Groups["day"].Value.Trim();

string ms = m.Groups["month"].Value.Trim();

string ys = m.Groups["year"].Value.Trim();

SentDateStr = ToDate(ds, ms, ys);

}

// check for "From" line

m = FromRX.Match(Line);

if (m.Success)

{

SenderName = m.Groups["From"].Value.Trim();

}

}

// Write last letter to XML

if (!string.IsNullOrEmpty(RecipientName))

{

NumOfLetters++;

OutputXml += OneLetterXML(NumOfLetters, SenderName, SentFrom, DateSent, RecipientName, RecipientLocation, LetterText, SentDateStr);

}

OutputXml += "</teiCorpus>";

System.IO.File.WriteAllText(DstFileName, OutputXml);

}

}