

על הקורס

- תכנות מערכות מבזירות
- הנושא של עיבוד big data על תשתית חישוב בענן, תחום מתפתח, חינני, אף אופנתי (במחקר, תעשייה)
- עבודה על שירות הנקרא AWS. תקבע חינם לכל סטודנט.
- ○ התמודדות עם שינויים בתכנים
 - עבודה מהבית (ואף עם Windows)
 - התמודדות עצמית, תמייכה של משה
 - הדוגמאות בקורס מתחום 'שימוש שפה טבעית' 'חובות הקורס':
 - 40% מבחן
 - 60% עבודות. 3 עבודות: 20,20,42
- .1. מבוא

1.1 מוטיבציה

1. עיבוד כמות גדולה של מידע

נדרש לכתוב תוכנית הסופרת כמה פעמים מופיעה כל מילה בראשת (מידע סטטיסטי שכזה הינו חינני עבור אלגוריתמים שונים של עיבוד שפה) לשם פשוטות, נניח כי כל קבוצי 'הראש' מאוחסנים כבר על שרטוי אחסון שונים ברחבי העולם (נכנה כל שרטת שכזה data-point).

ונסה לתאר תוכנית זו בຄלים שיש לו היום:

```
For each data-point
  For each doc in data-point
    Download doc
    for each word in doc
      inc word count in table
```

חסרונות העיצוב הנוכחי:

- תקשורת: יש להוריד את כל הרשות למחשב שMRIIZ את התוכנית
- זמן חישוב: גם ללא תקשורת, נדרש זמן רב ביותר לעبور על הקבצים המקומיים
- ניתן למקבל את החישוב ע"י ת'רידים, אך זה מוגבל במחשב בודד.

- נבחן מודל אחר עבור משימה זו.
הרעין המרכז: במקום לשלוח את הנתונים (קבצי הטקסט) לחישוב (המחשב שMRIIZ את התוכנית), נשלח את הקוד לכל שרת המאחסן מידע.

```

For each data-point
Send and apply:
{
  For each doc in [local] data-point
    for each word in doc
      inc word count in [local] table
}
Aggregate local tables

```

תקשורות: שליחת קטע קוד ציר לכל השרתים, שליחת טבלאות הסיכון המקומיות מכל שרת (לא את הטקסטים עצמם).
מקובל: רמת המקובל תואמת לכמות המידע אותו יש לעבוד. ככל שיש יותר קבצים בראשת, קטעי הקוד ירוצו על יותר מחשבים.

ב. טיפול במספר רב של לקוחות
נדרש לכתוב אפליקציה עבור רשת חברותית עם מספר לקוחות גדול (פייסבוק, טוויטר, יוטיוב...)

ניתן למש בקלות את הפוטווקול של טוויטר (התחברות, הוספה עוקב, שליחת הודעה), לדוגמה, בעזרה התבנית הכלכלית של הריאקטור שנלמדה בתכונות מערכות.

החיסון המרכז: לא ניתן לטפל במספר רב של לקוחות בתהילך אחד.
גם במחשב עם חומרה אינטואטיבית, לא ניתן לטפל ביותר מאשר אלפי לקוחות – The C10K Problem –
בשל המוגבלות של מערכות הפעלה הקיימות (מגבלה על מספר הקבצים/סוקטים הפתוחים, מגבלה על מספר המנעלים, וכו')
 █ כתוב את מערכת הפעלה מחדש
עדין מוגבל (אין באמת חומרה אינטואטיבית)
 █ נהפוך קבוצת מחשבים למחשב לוגי אחד, בעזרת מערכת הפעלה ייודית. כמות המחשבים בקבוצה תיקבע דינמיית על פי מספר הלוקוחות המוחברים לשרת כרגע.

לא עוסק בארכיטקטורה זאת, אך נזכיר על הנזודות הזהות בשתי הביעות:
 - האלגוריתם הלוגי פשוט (ספרת מילימ, מימוש פרוטוקול של מספר קטן של סוגים הודעות)
 - רכיב של משהו שגדל (כמות הנתונים לעיבוד, כמות הלוקוחות לטיפול)

- הפתרון המוצע מבוסס על ארכיטקטורה המשתמשת באופן דינامي במלאי של מחשבים בהתאם
לגודל (הנתונים, הליקוחות)

ג. מקובל אלגוריתם כבד עם מעט נתונים

המקורה הקלאסי-מסורתית של עיבוד מבוצר.

2.1. הצורך בעיבוד כמות גדולה של מידע

האם בכלל צריך לעבד כמות גדלות של מידע?

מסתבר שכן...

צורך אפליקטיבי

דוגמאות:

- מנוע החיפוש של גוגל

ו יש לעבור על הטקסטים בראשת כדי לבנות את האינדקס של מנוע החיפוש (טבלה המציגת עבור כל מילה את רשימת הדפים בראשת בהן היא מופיעה)

ו יש לעבור על כל הטקסטים בראשת כדי לקבוע את מידת החשיבות של כל דף בראשת, או את מידת ההתאמאה של מילה לדף זה, לשם דירוג תוצאות החיפוש.

- עיבוד נתונים ל��חות (נניח של חברת סולור, או של משתמש בראשת חברותית)
- ריגול תעשייתי או מדיני

- עיר חכמה

- עיבוד אירוע חדשתי מתגלאל ע"פ הרשותות החברתיות

- עיבוד תמונות ממצלמות, מרשומות חברותיות

מדובר בכמות גדלה ביותר של מידע לעיבוד.

בכל המקרים הנ"ל, המידע קיים, מאוחSEN, וכן קיימת תשתיית חומרה לעבד אותו (קלאסטרים עצומים של מחשבים), נדרשת רק ארכיטקטורה ותבנית תכונות המאפשרים לחבר ביניהם.

צורך מחקרי

בשיטת מחקר קלסיות, אוסף נתונים על הבעיה (תצפיות כוכבים, התבוננות במיקרואיסופ, מדידות פיזיקליות, בלשן שאוסף טקסטים...), ולאחר מכן מנהל מנתחים את התוצאות, 'אנליזה', ומנסים למצוא את הכלל, את החוק, את התבנית. מציעים 'מודל' שסביר את התופעות הנצפות.

בשיטת מודרנית, כמוות המידע הנცפית היא עצומה (טלסקופ החדש בצלילה מייצר מדי יום מיליון צילומים של השמיים ברזולוציה מטאורפית, מאגרי גנים DNA/RNA, פרויקט שחזור המפעץ הגדול בז'נבה, מאגר כל הטקסטים שנכתבו אי פעם,...), ופותחו אלגוריתמי למידה המוצאים תכניות בכמות מידע עצומות אלו.

כדי לישם מתודה מחקרית זו, נדרש שוב להריץ את האלגוריתמים הקיימים, על המידע הנוכחי, בעזרת מלאי מחשבים זמינים.

ספציפית עבור מדעי המחשב, קיימת טענה כי כאשר הקולט לאלגוריתם הינו עצום, האיכות של האלגוריתם לא רלוונטית לאיכות התוצאות. במקרים אחרים, כאשר יש כמות גדולה של מידע, אלגוריתם נאיבי ופשוט יגיע לאותן תוצאות כמו אלגוריתמים מתחכמים ומוסובכים. במקרה אחר, המוקד במדעי המחשב אינו פיתוח אלגוריתמים מדויקים יותר אלא איסוף מידע עבור הלמידה.

דוגמאות

1. בניית מסווג, classifier

סוג הינו מכונה הידועה לענות על שאלה מסוימת (נניח שאלת בינהית) ביחס לאובייקט נתון.
לדוגמא:

- o מסווג מקבל פרי, ועונה על השאלה: 'האם זה תפוח?'
- o מסווג מקבל דואר, ועונה על השאלה: 'האם זה דואר זבל?'

כיצד בונים מכונה שכזו?

הגישה של Machine Learning :

- o איסוף דוגמאות
- נבחר פרוטות שונים, ונבקש מבן אדם לציין עבור כל פרי האם הוא תפוח או לא.
- נבחר מיללים שונים, ונבקש מבן אדם לציין עבור מיל האם הוא ספאם.

o ייצוג האובייקט של השאלה על ידי אוסף מאפיינים

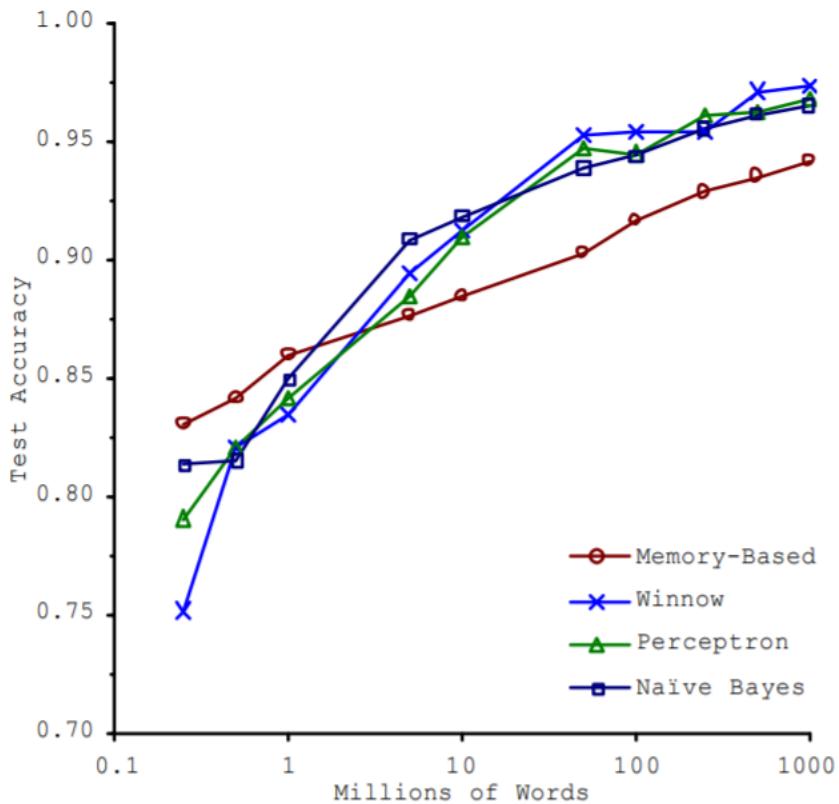
- כיצד מייצגים תפוח (אי אפשר לשולח לפונקציה בקוד 'תפוח')?
יש אינספור מאפיינים לתפוח – נבדוק עם מומחה בתחום מה הם המאפיינים המרכזים/החשובים להחלטה האם פרי מסוים הוא תפוח: גודל, מרקם (חלק,BINONI, מחוספס), צבע (ירוק, אדום,...), משקל, ...
באופן ניתן לייצג כל פרי ע"י וקטור של מאפיינים.

- כיצד מייצגים דואר?
מיללים, הקטגוריות של המילים (פועל, שם עצם, תואר, שם פרטי...), התפקיד התחבירי של המילים (נושא, נושא, מושא)

- o אלגוריתם למידה
אלגוריתם מקבל איסוף דוגמאות - כאשר כל דוגמא היא אובייקט (המיוצג ע"י אוסף מאפיינים) ותשובה ידנית לשאלת – ולומד/מכיל/מוצא את השיטה כיצד לענות באופן כללי לשאלת צו עבור אובייקטים חדשים.

מבין שלושת השלבים/החלקים של בניית המסוווג, נראה כי אלגוריתם הלמידה הוא החשוב, אחריו הגדרת המאפיינים, כאשר בניית אוסף הדוגמאות נחשב לחלק הפשטוט ביותר.

[Scaling to Very Very Large Corpora for Natural Language Disambiguation](#), ACL 2001) הראו כי עברו בניית מסוווג לדואר זבל, כל האלגוריתמים מגיעים לאותן תוצאות כאשר יש אוסף עצום של דוגמאות.



2. מענה לשאלות (Question Answering)

נתונה שאלה: מי רצח את לינקולן?
יש לחת תשובה (ג'ון ויליאם בוט')

כדי לבנות מערכת שיכזו, יש אלגוריתמים מתחכמים המבוססים על מאגרי מידע עצומים (אנציקלופדיות, מיליוןים, בסיסי נתונים), ועל כלים עיבוד שפה.

דוגמה: [במחשב ווטסון שניצח בתוכנית אלף האלופים של "ג'פארדי" \(הדיביטר של IBM\)](#)

lain ([An Exploration of the Principles Underlying Redundancy-Based Factoid Question Answering](#), 2007) הראה כי בהינתן כמות עצומה של טקסט, ניתן להסתפק באlgorigthm הפשטוטוט יותר לבעה זו:

מי רצח את לינקולן?

נחפש במאגרי הטקסט העצומים את המחרוזת 'רצח את לינקולן', ונניקה את המילה שמשמעותה היכי הרבה פעמים לפניה.

בשילוב זה, הטכניקות לייצור מהירה של אוסף דוגמאות גדול התפתחה מאוד טכנולוגית בשנים האחרונות, ואף הפכה לתחום מדעי עצמאי.

לדוגמה: Amazon Mechanical Turk

לעתים, הדרך היחידה ללמוד נושא מסוים היא על ידי עיבוד גדול של מידע: לדוגמה, לימוד אוטומטי של שפה. אם רוצים לעמוד על כל השינויים המתורחשים בשפה, יש לעבד כל הזמן טקסטים מתחדשים.

כיצד מלמדים מערכת ממוחשבת שפה אנושית?

- לימוד מונחה חוקים (דומה ללימוד אנושי של שפה זרה)
נדין למחשב את כל חוקי השפה: מה המילים בשפה (לקסיקוגרפיה), מהו מבנה המילים בשפה (מורפולוגיה), מהו מבנה המשפט (תחביר), מהו המשמעות של המשפט (סמנטיקה) כדי לנתח טקסט, המערכת למשא 'מקומפלט' את הטקסט לאור החוקים.

חסרונות: מי יודע את החוקים? כיצד ניתן לפרטם? החוקים משתנים כל הזמן...

- לימוד ע"פ הסתברויות מהtekstyim עצם (דומה ללימוד שפט שם אצל בני אדם)
בשיטת לימוד זו מknim למערכת הממוחשבת 'אינטואיציה', יש דברים שמסתברים ויש דברים שלא.

* אני ללכת הביתה

נראה מקרים שונים, שבהם הלימוד האוטומטי דורש עיבוד חוזר ונשנה של טקסטים חדשים (=ביבג DATA)

מבנה הפעיל: 'דני משכיר דירה' - מיהו המשכיר? המשאל?

'X משכיר את'
'X משכיר ל'

הוא יIRON, זה לא זמן טוב לכתוב שירים: [גרסה מוקדמת, גרסה מאוחרת](#)

מעמד התחליות: ה' הידעה, ב' היחס

ילד, הילד, חמוד, החמוד
בשביל, א' הבשビル, א' הלא
"הלאו דזוקא בלתי מצוי"
"הבשビル של' נעשה חשוב יותר"

בית, בָּבִית, בְּבִית

צורת הבינוני

דני ספר אוטיות [אטמול]
דני סופר אוטיות [כעת]
דני יספר אוטיות [מחר]

סופר

דני הזכיר לאכול
דני מזכיר לאכול
דני יזכיר לאכול

מזכיר

שוליה הזכרה לאכול
שוליה מזכירה לאכול
שוליה תזכיר לאכול

זיכרון

מנגנון זה מייצר כל הזמן שמות חדשים מפעלים בצורה ביןוני: כותב, מטפס, לוד.

'lod Shl'

אנקדוטה: [אני פרפר](#), אביב גפן

אני פרפר
שהיה פעם גולם

از תפסיק בCONFIG לאחיז
את יכול לאروع

מתרגם הסתברותי כמשמעות תרבות על בסיס קורפוס עצמוני:

I wash the car / I wash the floor / I am very nervous / I am very hysterical
אני שוטף את האוטו / אני שוטפת רצפה / אני מאוד עצבני / אני מאוד היסטרית

1. בכמה DATA מדובר?

2EB: 2000

2EB: 2011 ליום

EB 15: 2014 ליום

... : 2021

How much data?

- Modern applications use massive data:
 - Rendering 'Avatar' movie required >1 petabyte of storage
 - eBay has >6.5 petabytes of user data
 - CERN's LHC will produce about 15 petabytes of data per year
 - In 2008, Google processed 20 petabytes per day
 - German Climate computing center dimensioned for 60 petabytes of climate data
 - Google now designing for 1 exabyte of storage
 - NSA Utah Data Center is said to have 5 zettabyte (!)
- How much is a zettabyte?
 - 1,000,000,000,000,000,000 bytes
 - A stack of 1TB hard disks that is 25,400 km high

© 2011-14 A. Haeberlen, Z. Ives

University of Pennsylvania

1.3 תוכנית הקורס

- מבוא
- תשתיות
 - o חומרה, AWS [תרגיל 1]
 - o סביבת ריצה, Hadoop
 - תבנית Map-Reduce
 - o דוגמאות בסיסיות
 - עיצוב 'אלגוריתמים' בתבנית Map-Reduce
 - o מודל שפה
 - o Join
 - o בעיות למידה
 - תיוג חלקו דבר (לימוד אוטומטי לא-מנחיה)
 - ניתוח תחבירי (אימון מסוווג מונחה)
 - סיווג מסמכים
 - מידול נושאים (רשתות נירוניות)
 - [תרגיל 2]
 - o בעיות למידה
 - תיוג חלקו דבר (לימוד אוטומטי לא-מנחיה)
 - ניתוח תחבירי (אימון מסוווג מונחה)
 - סיווג מסמכים
 - מידול נושאים (רשתות נירוניות)
- [תרגיל 3]
- o מנוע חיפוש
 - o כיצד לדרג את תוצאות החיפוש (עיבוד מבוצר של גרפ)

2. תשתיות - Cloud Computing

2.1 מבוא

- שבו של החישוב המקביל
 - o באופן די ברור עדיפה כמות גדולה של מחשבים פשוטים (out) מאשר כמות קטנה של מחשבים חזקים מאוד (up)
 - אנרגיה (חשמל, קירור)
 - תחזוק
 - ...
- מה זה קלואוד-קומפьюטיניג
 - o שכירת שירותים, על פי הצורך כרגע

בailo שירותים מדבר?

o IaaS (Infrastructure As Service) שרותי חומרה: מעבדים, זיכרון, אחסון, תקשורת

o PaaS (Platform As Service) מה 'מוצקן' על החומרה: מערכת הפעלה, JVM, בסיס נתונים...

o SaaS (Software As Service) שכירתי שימוש בתוכנה שרוצה בקהלאוד Web Services

- וירטואלייזציה

- המחשבה פיזית של אתר שצד - במצגת. הניסיון של מיקרוסופט [להטמין דата-סנטר מתחת למים](#).

2.2 שירותי החישוב של AMAZON

Amazon Web Services (AWS)

1. שירותי חומרה ופלטפורמה

(Amazon Elastic Compute Cloud (EC2

שכירתי חומרה עם סט של תוכנות מותקנות:

- בחירת תצורת חומרה ([instance type](#))
- בחירת `image` (מה 'מוצקן' על החומרה, 'תכולת כונן ami', 'C')
- בחירת כמות מחשבים נדרש
- אבטחה
- גמישות
- בקרה
- ...

[AWS SDK for Java](#)

[API Reference](#)

[AWS SDK PROJECT ON GITHUB](#)

[AWS SDK JAVA V2 EXAMPLES](#)

דוגמת קוד: [CreateInstance](#)

```

Ec2Client ec2 = Ec2Client.create();
String amild = "ami-076515f20540e6e0b"; // Linux and Java 1.8

RunInstancesRequest runRequest = RunInstancesRequest.builder()
    .instanceType(InstanceType.T1_MICRO)
    .imageId(amild)
    .maxCount(1)
    .minCount(1)
    .userData(Base64.getEncoder().encodeToString(
        /*your USER DATA script string*/.getBytes()))
    .build();

RunInstancesResponse response = ec2.runInstances(runRequest);

List<Instance> instances = response.instances();

```

אפשרויות נוספות:

- הוספת תגים למחשבים השונים, כדי לאפ"י כל לפי הצורך (נניח מי 'מנהל' ומיל' עובד')

```

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceId)
    .tags(tag)
    .build();
ec2.createTags(tagRequest);

```

- מנגנונים המבטיחים גישה מאובטחת למידע

keyName - public key cryptography to encrypt and decrypt login information, [more](#).

```
.keyName(/*your KEY.PEM name*/)
```

iamInstanceProfile – AWS role define the permissions the instance will have, [more](#).

```
.iamInstanceProfile(IamInstanceProfileSpecification.builder().arn(/* your ROLE ARN */))
```

- קביעת ברירת המחדל עבור פקודת shutdown

instanceInitiatedShutdownBehavior – what the instance would do (stop or terminate) then the cli command "shutdown -h now" preform.

```
.instanceInitiatedShutdownBehavior(/*"terminate" or "close"*/)
```

2. שירותים אחסון

o אחסון רגיל במערכת קבצים - 'דיסק-און-לי' Elastic Block Storage (EBS)

- המידע הנשמר נגיש רק מחשבי EC2
- המידע הנשמר נגיש רק מחשב אחד במקביל
- ה EBS הוא 'כון' במערכת קבצים (ניתן לבצע mount), המידע הוא 'קובץ', ניתן לחזור אליו בהמשך.
- הגודל מוגבל
- יש עלות

o מאגר של אובייקטים הנitinנים לגישה ע"פ מפתח

(Amazon Simple Storage Service (S3

- ניתן לשמר אובייקטים (ובפרט קבצים) תחת מפתח נבחר
- רמת היררכיה אחת: buckets
- ניתן לגישה כמעט מכל דבר (בפרט מהרשת, http,...), אין צורך לגשת דוא"ל
- מחשב של EC2
- גודל כמעט לא מוגבל
- זול
- אין סמנטיקה של מערכת קבצים

דוגמא: [S3Operations.java](#)

```
S3Client s3 = S3Client.builder().region(Region.US_WEST_2).build();
s3.createBucket(CreateBucketRequest.builder().bucket("dsp241").build());
s3.putObject(PutObjectRequest.builder().bucket("dsp222").key("ass1").build(), new
File("assignment1.pdf"));
S3Object ass1 = s3.getObject.GetObjectRequest.builder().bucket("dsp222").key("ass1").build();
s3.deleteObject(DeleteObjectRequest.builder().bucket("dsp222").key("ass1").build());
s3.deleteBucket(DeleteBucketRequest.builder().bucket("dspp222").build());
```

o בסיס נתונים

אפשרויות:

- נריץ מחשב ב EC2, נתקין עליו תוכנה של MySQL (MySqlServer לדוגמה), נשמר את ה כל c. הריצה של מחשב EC2 אחר כר עם image זה מאפשר להריץ את

- שרת בסיס הנתונים (הנתונים עצם של בסיס הנתונים – בפרט הטבלאות – יישמרו ב EBS, CI S3 אינה מערכת קבצים, ומערכת ניהול בסיסי הנתונים מצפה לקבל מסלול לספריה במערכת הקבצים)
- נשתמש בשירות של בסיס נתונים ב cloud:
 - בסיס נתונים רלוֹצִיּוֹן (טבלאות, שדות וכו') - RDB
 - בסיס נתונים שאין רלוֹצִיּוֹן (Simple/DynamoDB) – no-sql db

3. תקשורת

ו תור משותף בין תהליכי, המאפשר תקשורת ביניהם

(Amazon Simple Queue Service (SQS

דוגמא: SQSExample

```
SqsClient sqs = SqsClient.builder().region(Region.US_WEST_2).build();
CreateQueueResult queueRes =
    sqs.createQueue(CreateQueueRequest.builder().queueName("dsp-queue").build());
String queueUrl = queueRes.getQueueUrl();
sqs.sendMessage(SendMessageRequest.builder().queueUrl(queueUrl).messageBody("hello
world").build());
List<Message> messages =
sqs.receiveMessage(ReceiveMessageRequest.builder().queueUrl(queueUrl).build()).messages();
for (Message m : messages)

    sqs.deleteMessage(DeleteMessageRequest.builder().queueUrl(queueUrl).receiptHandle(
        m.receiptHandle()).build());

sqs.deleteQueue(DeleteQueueRequest.builder().queueUrl(queueUrl).build());
```

3. סביבת הריצה עבור תבנית Map-Reduce – Hadoop

3.1 מבוא

נחזיר לתוכנית ספירת המילים, בה פתחנו את הקורס:

For each data point

Send & apply:

{

For each doc

For each word

Inc word count

}

Merge local counts

ברצוננו להריץ את התוכנית על התשתיות הענפה והגמישה של AWS.

אפשרי (כמו בתרגיל 1):

- ה"מנג'ר" ירץ את התוכנית הראשית, וכל "וורקר" יבצע את הפעולה על point-data-point נתון, כאשר התקשרות ביניהם תיעשה באמצעות SQS
- המנג'ר בוחן את כמות הקבצים לספרה במאגר, ומחליט כמה וורקר נדרשים.
- המנג'ר שולח בSQS לכל וורקר את ה point-data-point הוא עובד
- הוורקר סופר את המילים בקבצים ב point-data-point שלו, ומעלה לSQS את הטבלה המקומית של הספירה, ומודיע למנג'ר (בSQS) שהוא סיים.
- המנג'ר ממחה שככל הוורקר יסיום
- המנג'ר ממזג את כל הטבלאות המקומיות הנמצאות בSQS

אך דורש התיאחסות למגוון רחב של נושאים:

- איזה מחשב מעבד איזה קובץ (רצוי שהוא שמור לו כדי לחסוך בתקשרות)
- העברת הקוד לביצוע לכל מחשב (עבור המקרה הכללי), הרצת הקוד בכל מחשב
- העברת המידע אליו עובד כל מחשב למחשב זה ('מערכת קבצים מבודצת')
- ניהול שבירת הפלט המקומי של כל מחשב (שלא דרך גורם שלישי כמו SQS)
- כיצד מתבצע בסוף איסוף ומיזוג התוצאות המקומיות (מעבר ללוגיקה של פועלות המיזוג לשczemna), כיצד ניתן לזרז אותה בעילות ונוחות.
- סyncronization
- טיפול בניפויות מחשבים, שגיאות
- מידול כללי, טיפולים, ועוד

¶ נדרשת סביבת ריצה ייועדת עבור תבנית זו: Hadoop

עבור על שני חלקים מרכזיים בסביבה:

- מערכת הקבצים המבודצת
- מנגנון ההרצה של תוכניות Map-Reduce

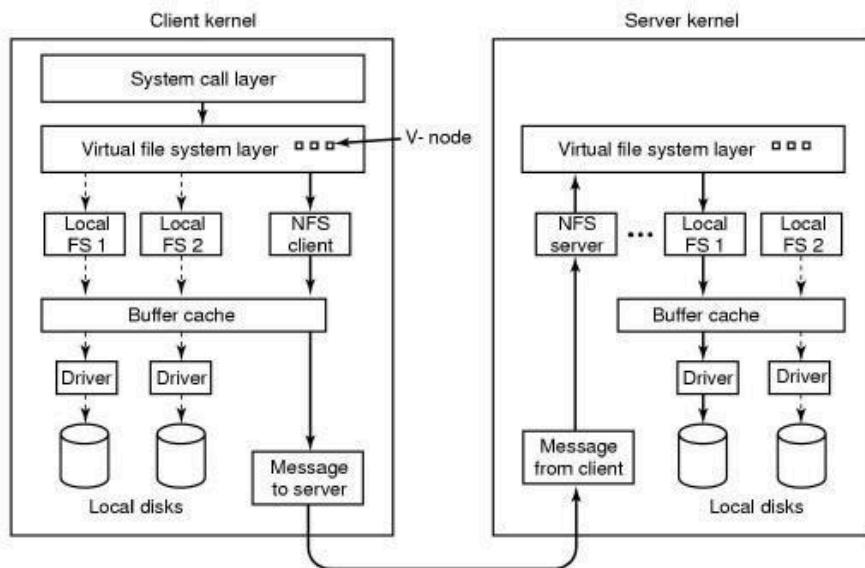
עבדה מול סביבה שכזו תשחרר אותנו מהטיפול בכל הabetים שצויינו לעיל, כך שנוכל להתמקד בלוגיקה של פעולה map (מה לבצע על המידע המקומי) ובלוגיקה של פעולה reduce (כיצד ממזג את התוצאות המקומיות).

3.2 מערכת הקבצים המבודצת

3.2.1 מבוא

מערכת קבצים מבודצת מאפשרת גישה לקבצים מרוחקים (המאוחסנים במחשב אחר בראשת) באותו אופן שבו אנחנו ניגשים לקובץ מקומי.

Network File System (2)



The NFS layer structure

38

חסרונות עברו המקרא שלנו:

- o נפילת מחשבי האחסון
- o זמן העברה כאשר הלוקה מרוחק מידי מקום האחסון

□ נדרשת מערכת קבצים ייעודית עבור התבנית בה אנחנו עוסקים.

אפיון השימוש במידע בקבצים בתבנית Map-Reduce:

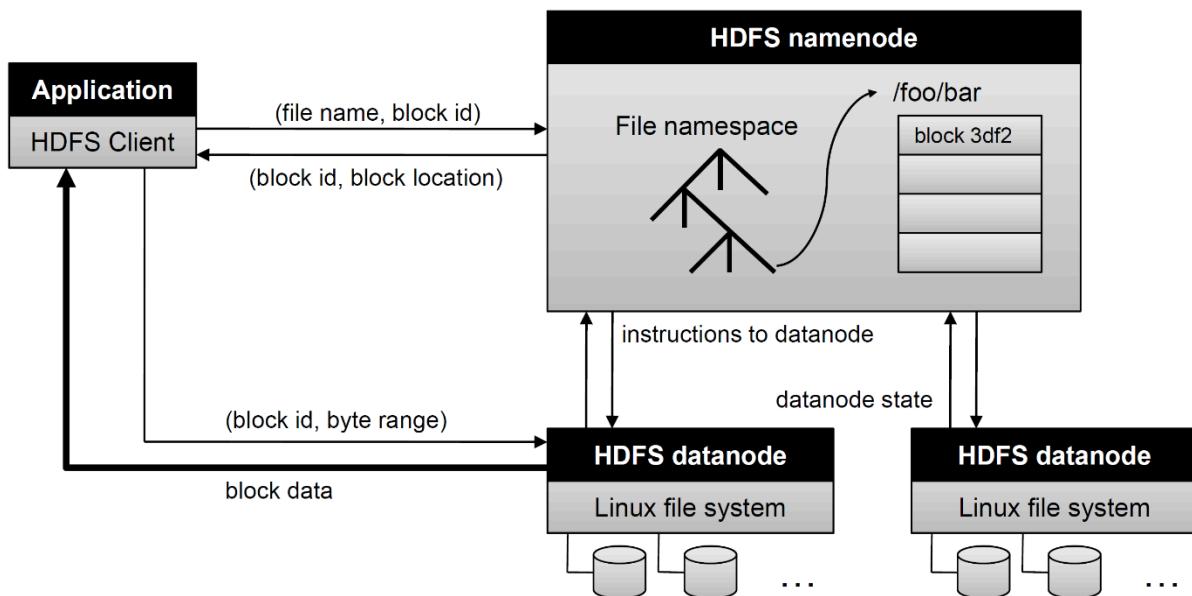
- קבצים גדולים
- המעביר על הקבצים הינו סדרתי
- או שקוראים קובץ או שמייצרים קובץ עם תוכן חדש (לא מעודכנים מידע בקובץ קיים)
- התמודדות עם נפילות

- תקשורת יעליה עבור העברת big data

?

- ו עותקים
 - באופן מוגנה, כל בלוק של קובץ נשמר בכמה עותקים במחשבים שונים.
 - כאשר מחשב המחזיק בעותק של בלוק נפל, קיימת אльтרנטיבה.
 - הלקוח ייגש לעותק הקרוב אליו ביותר.
- ו Immutability
 - לא ניתן לשנות קובץ. או שקוראים או שכותבים חדש.
 - משחרר את הצורך לסקירה/לנעל
- ו קבצים סדרתיים
 - הקבצים נשמרים מראש, כבירית מיוחד, באופן כמה שהוא יותר סדרתי, במבנה של רשומות מקודדות בינהירית כ `<key,value>`
 - העדפת קבצים גדולים (ללהן)
- ו שירותים מצומצמים - לא כל פעולה במערכת קבצים רגילה ממומשת בDFS

ארQUITקטורה



דוגמת קוד (באחר): תוכנית המקבלת כקלט ספריה במערכת הקבצים המקומיות, קוראת את הקבצים שורה ושרה, ושומרת אותם כקובץ מאוחד אחד במערכת הקבצים המבוזרת.

```
public class MergeFiles {  
  
    // Concatenates files from a given input local directory into a given local output file  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        FileSystem hdfs = FileSystem.get(conf);  
        FileSystem local = FileSystem.getLocal(conf);  
  
        String inDirname = args[0];  
        String outFilename = args[1];  
  
        //File outFile = new File(outFilename);  
        Path outFile = new Path(outFilename);  
  
        //OutputStream outstream = new FileOutputStream(outFile);  
        OutputStream outstream = hdfs.create(outFile);  
  
        PrintWriter writer = new PrintWriter(outstream);  
  
        //File inDir = new File(inDirname);  
        Path inDir = new Path(inDirname);  
  
        //for (File inFile : inDir.listFiles()) {  
        for (FileStatus inFile : local.listStatus(inDir)) {  
  
            //InputStream instream = new FileInputStream(inFile);  
            InputStream instream = local.open(inFile.getPath());  
  
            BufferedReader reader = new BufferedReader(  
                new InputStreamReader(instream));  
  
            String line=null;  
            while ((line = reader.readLine()) != null)  
                writer.println(line);  
            reader.close();  
        }  
        writer.close();
```

```
}
```

3.3 מנגנון ההרצה של תוכניות map-reduce

נזכר תחילה בפעולות Reduce, Map בתוכנות פונקציונאל:

```
map((x) => x*x, [1,2,3]);
  ↪ [1,4,9]

reduce((x,y) => x+y,0,[1,2,3])
  ↪ 6

reduce((x,y) => x+y,0, map((x) => x*x, [1,2,3]))
  ↪ 14
```

יש להכליל פונקציות אלו למקורה המבוזר:

- כאשר אין זיכרון משותף, וכו'
- מידול, טיפוסים, הפשטה של קריית הקלט...

¶ כדי שכבר ציינו, נדרשת לשם כך סביבת ריצה ייעודית

נכתב תחילה תוכנית map-reduce לדוגמא. תוכנית הסופרת את מספר מופיעים המילים השונות בקובץ.
טיקסט רבים הניתנים בקלט.
התוכנית WordCount באתר הקורס.

נבחן כיצד ריצה התוכנית שכתבנו ע"י Hadoop במקורה המבוזר:

- הג'וב שהגדכנו מכיל את ההגדרות הבאות:
 - o מחלקת ה Mapper, הקובעת את יחידת המידע לפעולה המקומית, ואת הפעולה המקומית (מספר שורה ותוכן שורה בקובץ קלט, ספירת המילים בשורה הנתונה תוך ייצור פלט של זוגות <מילה, 1>)
 - o מחלקת ה Reducer, הקובעת כיצד מציגים לערך אחד את הערכים השונים שנוצרו ע"י ה Mappers עבור מפתח אחד נתון (כיצד מציגים את מספרי המופיעים של מילה נתונה לסכום אחד)
 - o רשימת קבצי הקלט (במערכת הקבצים המבוזרת)
 - o ספירת הפלט (במערכת הקבצים המבוזרת)
 - o האופן שבו מחלקים קבצי הקלט ליחידות קטנות יותר*, והאופן שבו מומרים יחידות אלו לרשימה הזוגות V-K המהווים את הפרמטרים של מתודת ה InputFormat (map).

- ו האופן שבו נכתב הפלט לקובץ. (OutputFormat)

*קבצי הקלט מפורקים ליחידות קטנות יותר. אופן פירוק מוגדר במחלקה InputFormat. מחלוקת זו כוללת מתודה בשם getSplits המתקבלת את רשימה קבצי הקלט ומחזירה רשימה של חלקים קבצים המכונים Splits (שיטת חלוקה סטנדרטית תבוסס על גודל ה-Split) הג'וב שהוגדר מוגש לסייעת Hadoop לבצע.

הארכיטקטורה של יחידת ההרצה/הביצוע של Hadoop מורכבת משני סוגים של תהליכיים/ת'רדים:

- מנג'ר ה-Job, ה"מנג'ר" של מנגנון ההרצה
- TaskTracker, ה"עובד"ים של מנגנון ההרצה

נבחן את דפוא הפעולה של כל אחד מהם:

JobTracker

- בהינתן Job

- ו מחלק את קבצי הקלט ליחידות קטנות יותר של ספליטים, על פי מתודת `getSplits()` במחלקה InputFormat הנתונה בג'וב.
- ו מגדיר עבור כל Split בג'וב משימה לביצוע עבור TaskTracker. משימה זו מוגדרת ע"פ הנתונים הבאים:

- ה split עליי הוא עובד.
- מכשיר ההופך את המידע לרשימה של key-value ע"פ הנדרש במתודת `map`.
- מכשיר זה, המכונה RecordReader, ניתן על ידי מחלוקת ה-InputFormat (`createRecordReader`)
- מחלוקת ה-Mapper (מה לבצע על כל key-value)
- ו השמת כל משימה לביצוע ע"י ה-TaskTracker הזמן והמתאים ביותר.
- ו המתנה לאישור ביצוע המשימה ע"י כל ה-TaskTrackers.
- ו החלטה כמה TaskTrackers נדרש עבור שלב הבא של מיזוג התוצאות.
- ו בחירת TaskTrackers לביצוע שלב המיזוג, עם משימת המיזוג עוברים.
- ו הначית ה-TaskTrackers של שלב ה-Map להעביר את ה key-value שהם יצרו למחשבים שנחקרו לבצע את מיזוג התוצאות. אופן הפיזור נקבע על פי המדיניות המומומשת במחלקה Partitioner (המתודה `getPartition`)
- ו בקשה מה TaskTrackers שנבחרו להרצת משימות ה-Reduce להתחילה לעבוד.
- במחשבים אלו מחכים כבר על זוגות ה-V-K שיצרו כפלט ע"י ה-Mappers ונשלחו אליהם.
- כל אחד מה TaskTrackers מקבל גם מה JobTracker את רשימת המיזוג שהוא נדרש לבצע (מחלקה Reducer שהוגדרה בג'וב), וכן את האופן שבו יש לכתוב את הפלט הסופי לקובץ הפלט (מחלקה OutputFormat שהוגדרה בג'וב).
- ו מכחח שככל ה-TaskTrackers יסימנו את משימות המיזוג שלהם.

TaskTracker המבצע משימות Map

- מקבל שימוש לביצוע, הכוללת: Split, RecordReader (מכשיר המפרק את ה Split ל רשיימת key-value), אובייקט מטיפוס Mapper מבצע את הלולאה הבאה

```
mapper.setup(context);
while (rr.nextKeyValue())
    mapper.map(rr.getCurrentKey(), rr.getCurrentValue(), context);
    mapper.cleanup(context);
```

- עדכון ה JobTracker על סיום ביצוע השלב המרכזי של המשימה. והמתנה להנחיות על השלב האחרון של המשימה: כמה Reducers נדרשים.

- Shuffle: שילוח key-value שנוצרו בהפעולות מתודת ה map למחשבים שיבצעו את מיזוג התוצאות (ע"פ המדייניות המוגדרת בחלוקת ה Partitioner הניתנת בג'וב)

- TaskTracker המבצע משימות Reduce במחשב של ה TaskTracker יש (קובץ) עם רשיימת key-value שהתקבלו כפלט של TaskTrackers משלב ה Map.

- Tasktracker מקבל אובייקט מטיפוס Reducer המגדיר את פעולה המיזוג הנדרשת (מימוש מתודת ה reduce) הפעולה שהוא מבצע

o Sort

- ארגון חדש במבנה נתונים יעיל וקומפקטי של רשיימת הזוגות:
 $\langle \text{כסא}, 1 \rangle$
 $\langle \text{שולחן}, 1 \rangle$
 $\langle \text{כסא}, 1 \rangle$



{ שולחן : [1],
 כסא : [[1,1]] }

- מיון השורות בטבלה ע"פ ה compare של מחלוקת המפתח.

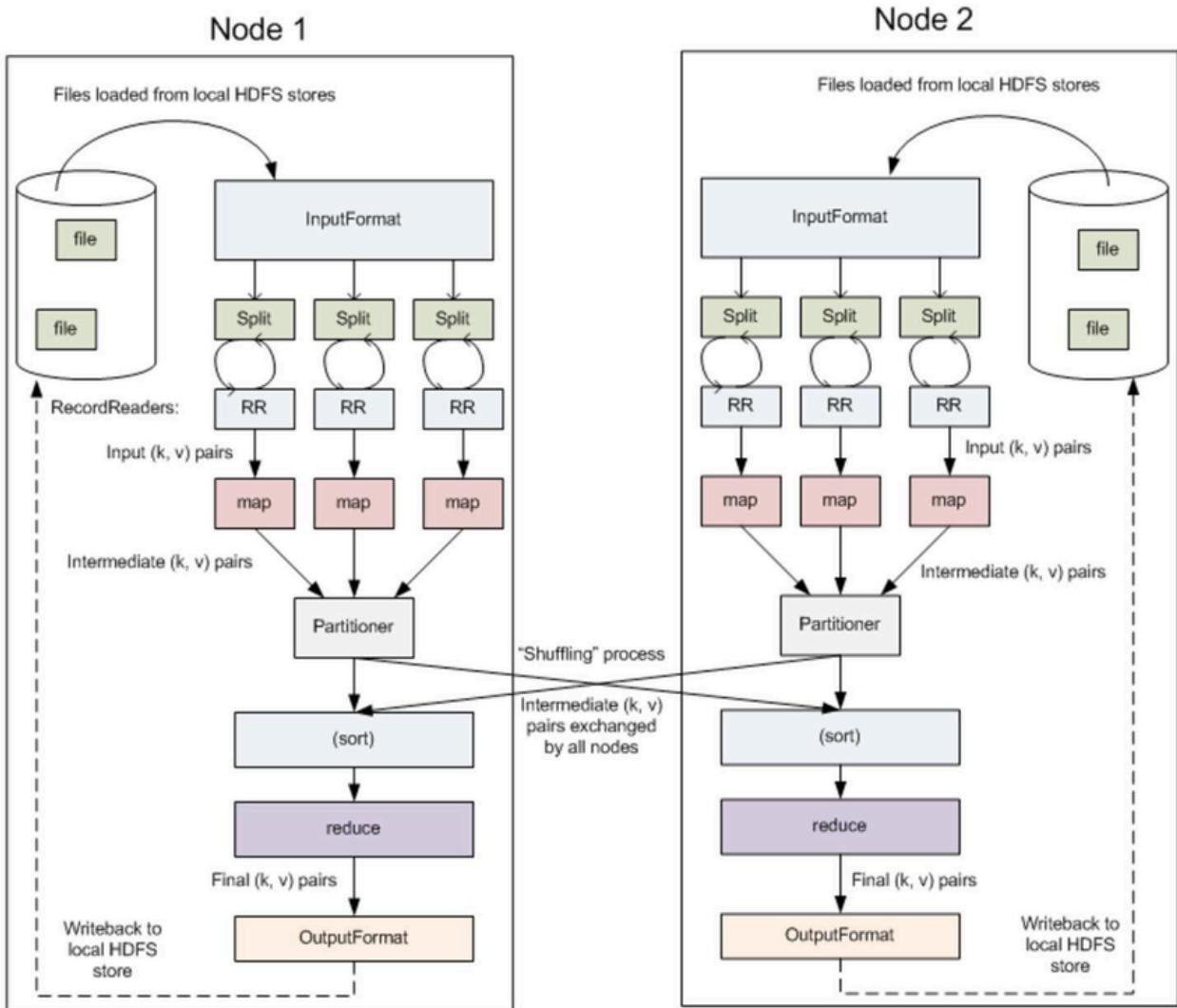


{ כסא : [1,1],
 שולחן : [1] }

o ביצוע הקוד הבא (על הטבלה המפתחות והערכים 'table':)

```
reducer.setup(context);
for (Entry entry : table.entrySet())
    reducer.reduce(entry.getKey(), entry.getValue());
reducer.cleanup(context);
```

- הפלט (רשימת key-value) שכל משימת Reduce מייצרת נכתב לקובץ פלט בספריית הפלט המוגדרת בג'וב, ע"פ הפורמט המוגדר בחלוקת ה OutputFormat (הניתנת במסגרת הג'וב)



דוגמא

קלט: פזמון שירו של יהוא ירונ זוכרים:

zochrim.txt
 בבטן השכחה
 זוכרים אותן זוכרים
 בבטנה של אמא
 זוכרים אותן זוכרים

נניח כי getSplits החלטה לחלק את הקובץ לשני Splits:

.a

בבטן השכחה
זכרים אותו זוכרים

.b

בבטנה של אמא
זכרים אותו זוכרים

כלומר, רצים שני TaskTrackers במשימת ה Map, אחד על ספליט a, והשני על ספליט b.

TaskTracker1

מקבל מה RR את הזוג <1, "בבטן השכחה">
קורא למתודת ה map, המייצרת את הזוגות:

<בطن,>
<שכחה,>
מקבל מה RR את הזוג <2, "זכרים אותו זוכרים">
קורא למתודת ה map, המייצרת את הזוגות:
<זכרים,>
<אותר,>
<זכרים,>

TaskTracker2

מקבל מה RR את הזוג <1, "בבטנה של אמא">
קורא למתודת ה map, המייצרת את הזוגות:

<בطن,>
<של,>
<אמא,>
מקבל מה RR את הזוג <2, "זכרים אותו זוכרים">
קורא למתודת ה map, המייצרת את הזוגות:
<זכרים,>
<אותר,>
<זכרים,>

שלב הניתוב:

במחשב של TaskTracker1 יש (בקובץ זמן) את הזוגות הבאים:

<בطن,>
<שכחה,>
<זכרים,>
<אותר,>

<זכרים, 1>

במחשב של TaskTracker2 יש (בקובץ זמן) את הזוגות הבאים:

<בטן, 1>
<של, 1>
<אמא, 1>
<זכרים, 1>
<אותר, 1>
<זכרים, 1>

נניח כי ה JobTracker החליט שנדרשים שני TaskTrackers למיזוג התוצאות המקומיות.
נניח כי שיטת הניתוב (כלומר מימוש Method getPartition במחלקה Partitioner) שהוגדרה בג'יב, היא חלוקת המפתחות ע"פ האות הראשונה: א-כ, ל-ת.

□

TaskTracker3 קיבל משימה מיזוג עבור מילימ המתחילה בא-כ:

<בטן, 1>
<זכרים, 1>
<אותר, 1>
<זכרים, 1>
<בטן, 1>
<אמא, 1>
<זכרים, 1>
<אותר, 1>
<זכרים, 1>

Sort
{
אותר: [1,1]
אמא: [1]
בטן : [1,1]
זכרים: [1,1,1,1]
{

```
reducer.setup();  
reducer.reduce(1,1);  
reducer.reduce(1,[ ]);  
reducer.reduce(1,[ ]);  
reducer.reduce(1,1,[ ]);  
reducer.reduce(1,1,1,[ ]);  
reducer.cleanup();
```

<אותר, 2>
<אמא, 1>

<בטן, 2>
<זוכרים, 4>

זוגות אלו ייכתבו לקובץ פלט בספריית הפלט במערכת הקבצים המבוזרת שהגדכנו בג'וב, ע"פ
הפורמט שהוגדר במחלקה OutputFormat:

אותך 2
אמא 2
בטן 2
זוכרים 4

TaskTracker4 קיבל משימת מיזוג עבור מילימ המתחילה בל-ת:

<שכחה, 1>
<של, 1>

Sort
{
שכחה: [1]
של: [1]
{

```
reducer.setup();  
reducer.reduce(1, [שכחה,]);  
reducer.reduce(1, [של,]);  
reducer.cleanup();  
<שכחה, 1>  
<של, 1>
```

זוגות אלו ייכתבו לקובץ פלט בספריית הפלט במערכת הקבצים המבוזרת שהגדכנו בג'וב, ע"פ
הפורמט שהוגדר במחלקה OutputFormat:

שכחה 1
של 1

דוגמא קוד נוספת: תוכנית המבצעת ניתוחים ופיתוחים סטטיסטיים על נתונים חברת סלולר.

כמו בכל עיצוב של קוד, נקבע תחילת את הטיפוסים הנדרשים למתודות. במקרה זה נגדיר שני סוגי
טיפוסים: User, Action

כלומר, פרטי הלקוח, ופרטים של פעולה אחת שהוא ביצע.

- הגדרת המחלקות User, Action, -
- הגדרת מושגים שונים על נתונים המשמשים ופועולותיהם בתבנית Map-Reduce -
- o ייצור גרף מתקשר-نعمן (מי התקשר למי)
- o ייצור גרף נמען-מתקשר (עבור כל נמען, מי התקשר אליו)
- o ייצור טבלה המצינית כמה פעמים התקשרו לכל לקוח בכל שנה.
- o היסטוגרמת שייחות שנתית

- התאמת פורמט הקלט של חברת הסולולר לייצוג C Action,User על ידי מימוש RecordReader מתאים.

o מבוא

נבחן תחילה, כיצד מומש TextInputFormat בו השתמשנו בתכנית ה WordCount כזכור, המשק [InputFormat](#) כולל שתי מתודות למימוש:
 getSplits – קובעת את מדיניות חלוקת קבצי הקלט ליחידות של splits
 Key-Value – קובעת כיצד לקרוא split נתון כרשימה של createRecordReader

מימוש getSplits במחילה TextInputFormat מוגדר במחילה האב שלה [FileInputFormat](#)

מימוש [createRecordReader](#) במחילה TextInputFormat מבוסס למעשה על המחלקה [LineRecordReader](#).

- o הגדרת LineRecordReader, על בסיס [UserActionRecordReader](#) הקיים ב Hadoop (על בסיס ההנחה שנתוני לקוחות ופעולה אחת מרכזים בכל שורה בקבצי הקלט של החברה).
 - החברה נדרשת רק למשם את המתודות האבסטרקטיות, parseUser, parseAction, המקבלות שורה מהקלט ומחזירות אובייקט User/Action

הערה: בעיצוב הקוד, נתקלנו במקרים שבהם נדרש לשדרש שני ג'וביים של map-reduce. לדוגמה:

- o ספירת כמות השיחות לשנה עבור כל לקוח (GenerateRecipientCallsPerYear)
- o ייצרת היסטוגרמת שייחות לשנה, ע"פ נתונים מספר השיחות לשנה של הלוקוחות השונים (GenerateRecipientCallsPerYearHistogram)

איך ניתן לבצע שרשרת שכזה?

1. נרים שתי תוכניות שונות, בזו אחר זו, תוך התאמת ספוריות הפלט של הראשונה והקלט של השנייה.
2. נגידר ב main של התוכנית שני ג'וביים, האחד עבור המשימה הראשונה והשני עבור המשימה השנייה, ונגיש אוטם לסיבובו (job.waitForCompletion()) בזו אחר זה, תוך התאמת הקלט והפלט.
3. מנגנון [JobControl](#)
4. מנגנון "יעודי" ב Amazon Elastic Map-Reduce

חלק שני: עיצוב אלגוריתמים בתבנית Map-Reduce

בחלק זה נעבור על בעיות שונות (בעיבוד שפה), ונבחן כיצד לעצב מחדש את האלגוריתם הסדרתי שלhnן אלגוריתם מבוזר בתבנית Map-Reduce.

נحدد תחילה, אלו דברים נמצאים בשליטתנו ואלו דברים נקבעים על ידי הסביבה:

ו סביבה

- הין ירצו ה Mappers וה Reducers
- מתי יתחלו להתבצע המשימות השונות

ו מעצב הקוד

- הגדרת מהם המפתחות ומהם הערכיהם, וכן מה הטיפוסים שלהם
- הגדרת קритריון מיון המפתחות (compareTo)
- הגדרת קритריון הנינווב של זוגות הפלט של ה Mapper (getPartition)
- הגדרת קוד אתחול וסיום לכל מסימה (setup, cleanup)
- שמירת מצב בזיכרון של אובייקט ה Mapper/Reducer
- שכפול מידע
- וכמוון:ימוש מתודות ה- map, reduce

הקוד שנעצב יבסס על 'ארגון הכלים' הנ"ל.

4. מודל שפה

4.1 מוטיבציה

1. זיהוי דיבור (Speech Recognition)

קלט: ערז סאונד

פלט: טקסט של מה שנאמר

אפשרויות: הפעלת מחשב בעזרת קול (ניהול עם שיחה עם רובוט, Saya), מודיעין, תרגום סימולטני...

מימוש:

- ו דגימת גלי הקול לסדרת וקטורים עם מאפייני הציליל (מספרים) [הנדסת חשמל]
- ו המרת וקטורי הציליל להברות
- ו אני תוו לה של טים
- ו שרשור ההברות לכדי מילים

אני תולה שלטים

אני תולש שלטים

אני תולש לטים

...

?

☒

(אני תולה שלטים)☒

(אני תולש שלטים)☒

(אני תולש לטים)☒

...

чисוב הסתברות זו, המבוססת על מאגר גדול של טקסטים מכונה מודל שפה

.2. (Optical Character Recognition (OCR

קלט: תמונה עם טקסט

פלט: הטקסט בתמונה

אפליקציות: סריקת ספרים (עבור מנוע חיפוש, מערכות למידה. גугл-בוקס, האנזה הקהירית,
מגילות קומראן), מודיעין

בעיה: לעיתים יש כמה אפשרויות לעתיק את הטקסט המצלום

(פורטים על שפע)☒

(פורטים על שמע)☒

.3. ניבוי מילים (Word Prediction

תקשורת תומכת וחליפת

(קיפה חזק או | חלש)☒

(קיפה חזק או | חם)☒

...

.4. תרגום אוטומטי

* תרגום מונחה-חוקים:

- ניתוח של הרמות השונות של הטקסט בשפת המקור - מיללים, משפטים, משמעות □ עד כדי מבנה נתונים המיצג את משמעות המשפט באופן שאינו תלו依 שפה.
- ייצור של המשפט בשפת היעד מבנה הנתונים המיצג את משמעותו – על פי חוקי הקובעים איך מתנסחים מיללים ומשפטים בשפת היעד.

* תרגום סטטייטי:

mbased על הסתבריות, ובפרט ההסתברות של התרגום המוצע

Danny went home

...

(Danny הלך הביתה) P

(Danny הלך לבתו) P

...

□ מודל שפה הינו רכיב מרכזי ביותר בכל אלגוריתם של לימוד שפה.

4.2 חישוב הסתברות של רצף מילים

Maximum Likelihood Estimation (MLE)

$$P_{MLE}(W_1 \dots W_n) = C(W_1 \dots W_n) / N$$

Where N is number of word sequences of size n in the corpus

$$N / (Danny הלך הביתה) C = (Danny הלך הביתה) P_{MLE}$$

Where N is number of triple words in the corpus

חסרון: דليلות מידע. אם סדרת המילים לא הופיע אפילו פעם אחת, ההסתברות תהיה 0.

$$(Danny הלך הביתה שלשות ולא היה לו מפתח להיכנס) P_{MLE}$$

פתרונות: החלקה (smoothing) – שערוך הסתברות למשפטים שלא נאמרו (טור וייתור מה עלי הדיקן עבור משפטי שנאמרו)

לדוגמא: back-off

$$= (Danny הלך הביתה שלשות ולא היה לו מפתח להיכנס) P_{bo} +$$

$$\alpha_1 P_{MLE} + (Danny)$$

$$\alpha_2 P_{MLE} + (Danny הלך)$$

$$\alpha_3 P_{MLE} + (Danny הלך הביתה)$$

$$\alpha_4 P_{MLE} + (Danny הלך הביתה שלשות)$$

$$\alpha_5 P_{MLE} + (Danny הלך הביתה שלשות ולא)$$

$$\begin{aligned}
 & + (\text{דני הילך הביתה שלשום ולא היה}) \alpha_6 P_{\text{MLE}} \\
 & + (\text{דני הילך הביתה שלשום ולא היה לו}) \alpha_7 P_{\text{MLE}} \\
 & + (\text{דני הילך הביתה שלשום ולא היה לו מפתח}) \alpha_8 P_{\text{MLE}} \\
 & (\text{דני הילך הביתה שלשום ולא היה לו מפתח להיכנו}) \alpha_9 P_{\text{MLE}}
 \end{aligned}$$

Where $\sum \alpha_i = 1$

מטרתנו בפרק זה: לחשב נוסחאות שכאללה, ע"פ קורפוו נתון (מאגר טקסט גדול), באופן מבוזר – בתבנית Map-Reduce.

- נבחן בעיה זו בשלושה שלבים:
- o ספירת מילים בודדות
 - o ספירת סדרות מילים
 - o חישוב ההסתברות

4.3 ספירת מילים בודדות

מימשנו כבר תוכנית למספר מילים בודדות (WordCount) לשם נוחות, נציג תוכניות שכאללה בפסאודה-קוד:

Class Mapper

```

Method Map(lineId, line)
For w in line
    Emit(w,1)

```

Class Reducer

```

Method Reduce(w, counts)
    sum := 0
    for (count in counts)
        sum := sum + count
    Emit(w, sum)

```

חרטן: המאפר מייצר כמות עצומה של זוגות `<מילה, 1>`: בעית תקשורת גדולה, ארגון מיון ומייזג כבדים בReducer

□ local aggregation, לפני העברת הזוגות, לבצע סיכום מקומי (במחשב הנתון או בספליט הנתון) של התוצאות.

מימוש:

.1. שבירת מצב בזיכרון, חלק מלוגיקת הקוד שאנו ממשיכים בחלוקת ה Mapper.

Class Mapper

Method Initialize()

H := new Table

Method Map(lineId, line)

For w in line

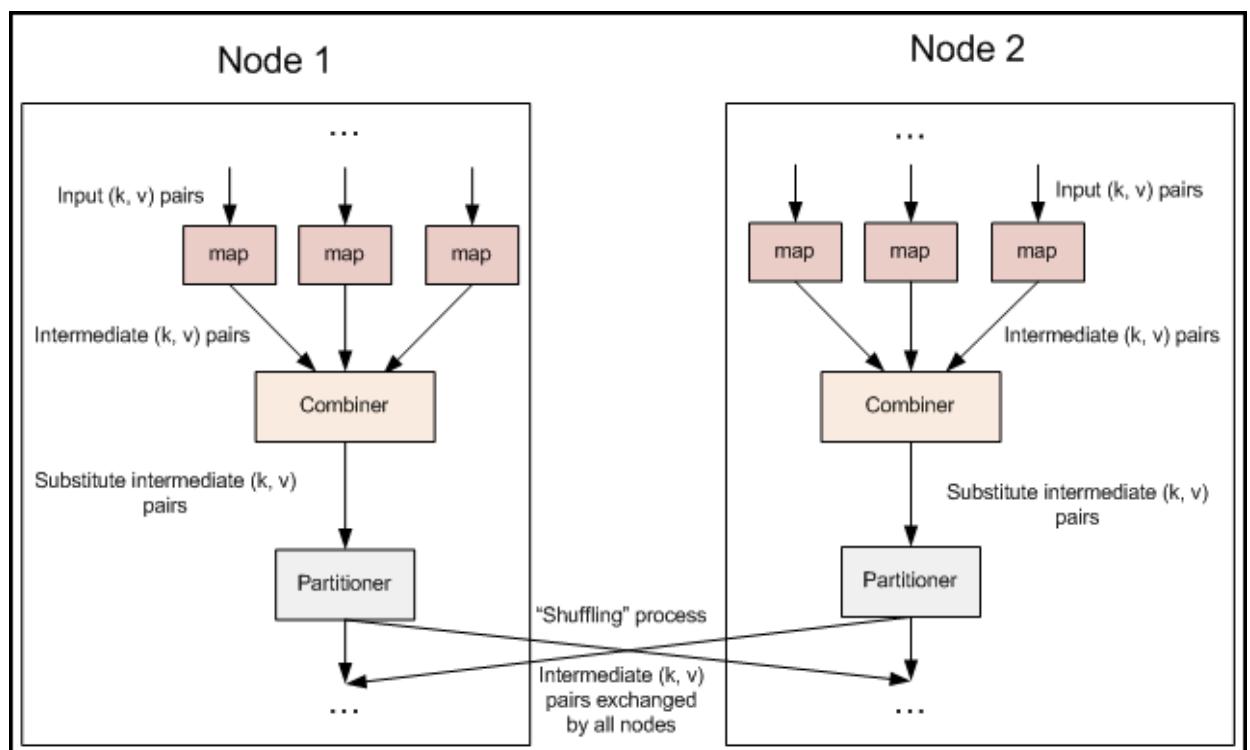
H{w} := H{w}+1

Method Close()

For <w, count> in Table

Emit<w, count>

2. כמנון של הסביבה: Combiner



Class Mapper

Method Map(lineId, line)

For w in line

Emit(w,1)

Class Combiner

```

Method Reduce(w, counts)
    sum := 0
    for (count in counts)
        sum := sum + count
    Emit(w, sum)

```

במקרה זה (ספרת מילימ), הקוד של מחלקת ה combiner זהה לחלוtin לקוד של מחלקת ה Reducer.
האם זה תמיד נכון?

דוגמא: חישוב ממוצע יומי של שיחות פר ללקוח

Class Mapper

```

method Map(userId, callPerDay)
    Emit(userId, callPerDay)

```

Class Reducer

```

Method Reduce(userId, callPerDays)
    sum := 0
    days := 0
    for (callPerDay : callPerDays)
        sum := sum + callPerDay
        days := days + 1
    Emit(userId, sum / days)

```

באופן זה, עבור לקוח ששווח 20, 15, 20 ישיחות נקל ממוצע של: 15 ישיחות ביום.

אם נשתמש בקומביינר שהקוד שלו זהה לקוד הרדיוסר, יתכן שקומביינר אחד יציג את 10, 15, 10 במחשב אחד ל-12.5, והשני את 20 ל-20, כך שהרדיאוסר יקבל 12.5, 20 ונקבל ממוצע סופי של 16.25 [ממוצע אין ממוצע של ממוצעים בהכרח]

¶ אם פעולות המיזוג אינה אסוציאטיבית וקומוטטיבית יש להגדיר קוד מיוחד לקומביינר.

Class Combiner

```

Method Reduce(userId, callPerDays)
    sum := 0
    days := 0
    for (callPerDay : callPerDays)
        sum := sum + callPerDay

```

```

days := days + 1
Emit(userId, >sum, days<)

```

Class Reducer

```

Method Reduce(userId, callsDaysPairs)
    sum := 0
    days := 0
    for (callsDaysPair : callsDaysPairs)
        sum := sum + callsDaysPair.first
        days := days + callsDaysPair.second
    Emit(userId, sum / days)

```

בעיה: שלב הקומביינר הינו אופטימיזציה, כלומר הוא יתקיים אם הדבר מתאפשר מבחינת הסביבה, אך לא בהכרח. במקרה שבו אין קומביינר, טיפוס הפלט של הממابر אינו תואם את טיפוס הקלט של הרדיוסר.

☒ יש לדאוג לפט אחד מבחינת הטיפוסים עבור המאפר והקומביינר

Class Mapper

```

method Map(userId, callPerDay)
    Emit(userId, >callPerDay, 1>)

```

Class Combiner

```

Method Reduce(userId, callPerDays)
    sum := 0
    days := 0
    for (callPerDay : callPerDays)
        sum := sum + callPerDay.first
        days := days + 1
    Emit(userId, >sum, days<)

```

יתרונות וחסרונות של שתי שיטות הקיבוץ המקומי: שבירת מצב בזיכרון – שימוש בקומביינר

- שבירת מצב בזיכרון עשויה להיות לא סקלובלית

לדוגמא, עבור ספירת מילים יש לשמר בזיכרון את כל המילים השונות בשפה. האם זה סקלבייל? כמובן, האם ניתן להניח שככל המילים השונות בשפה נכנסות לזכרון? האם הם יוכנסו לזכרון גם כאשר הקורפוס יגדל מאד? במקורה שלנו, השאלה היא: עד כמה גודל אוצר המילים השונות עם גידול מאגר הטקסט? יש לבדוק זאת מראש: תיאורטיבית או מעשית.

תיאורטיבית, לדוגמא: חוק Heap קובע את היחס בין גודל הקורפוס (מאגר הטקסטים) לבין גודל אוצר המילים (המילים השונות בקורסוס)

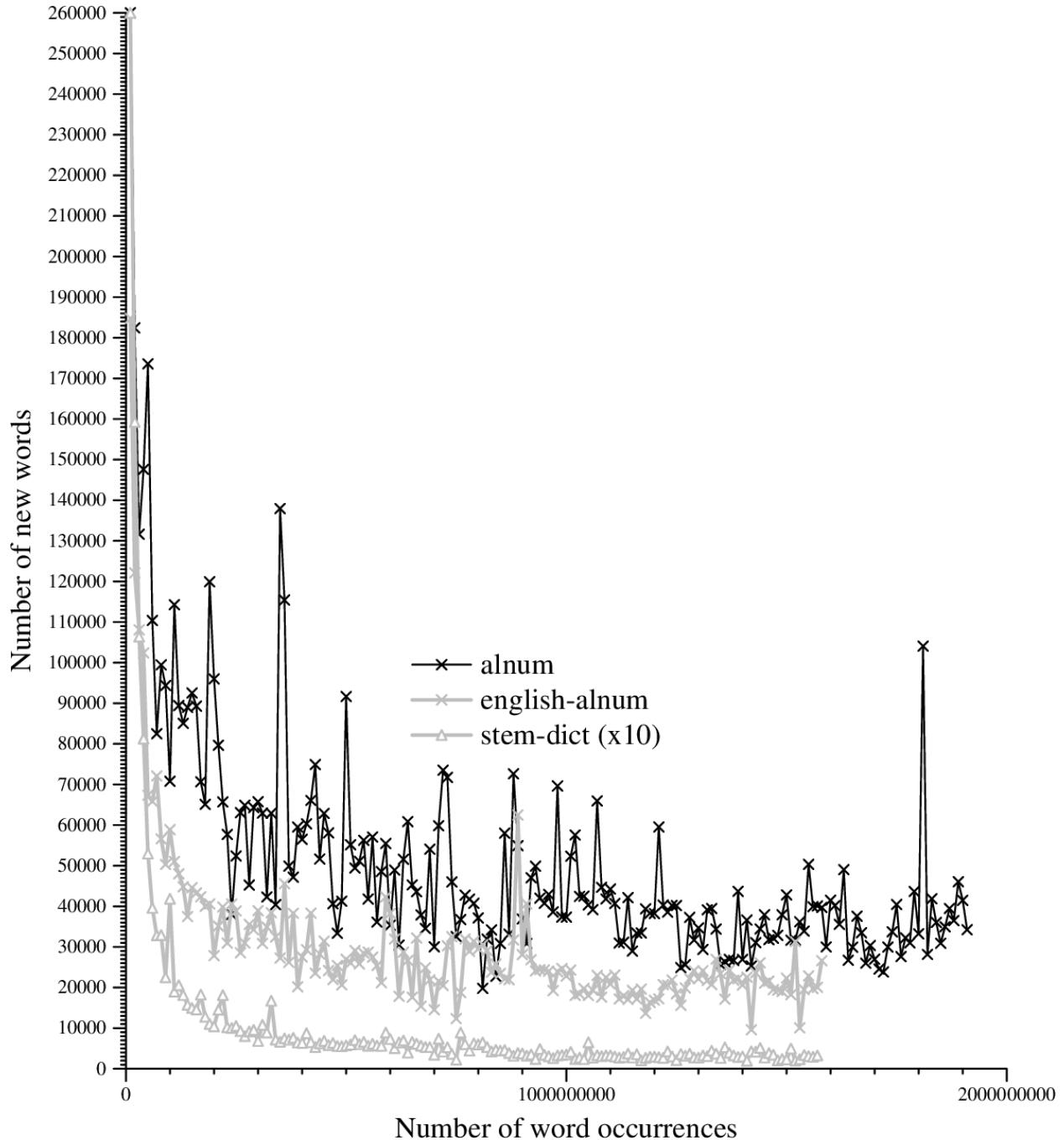
$$V = kT^b$$

כאשר V הוא מספר המילים השונות (אוצר המילים), T גודל הקורפוס, ו b, k פרמטרים.
 $30 \leq k \leq 100$, $b \sim 0.5$

מעשית, ניתן להריץ ניסויים על מאגרים שונים כדי לבדוק את קצב גידול המילים החדשנות ביחס לגודלו הקורפוס.

לדוגמא:

Searchable words on the Web, Hugh E. Williams, Justin Zobel, International Journal on Digital
[Libraries 2003]



מהן אותן מילים חדשות?

[Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis, Adler et al, ACL 2008]

Category	Examples	Distribution	
		Types	Instances
Proper names	'asulin (family name) 'a'udi (Audi)	אסולין אודי	40% 48%
Neologisms	'agabi (incidental) tizmur (orchestration)	אגבי ^{תזמור}	30% 32%
Abbreviation	mz "p (DIFS) kb "t (security officer)	מ"פ ק"ט	2.4% 7.8%
Foreign	presentacyjah (presentation) 'a 'ut (out) right	פראנציה אוט right	3.8% 5.8%
Wrong spelling	'abibba 'ahronah (springatlast) 'idiqacyot (idication) ryušalaim (Rejusalem)	אבייבאבאורה אידייקציות ריישלאים	1.2% 4%
Alternative spelling	'opyynim (typical) priwwilegyah (privilege)	אופיינים פרויוילגיה	3.5% 3%
Tokenization	ha "sap (the"threshold) 'al/17 (on/17)	ה"סף על/71	8% 2%

בגישה הקומבינר, שאליה זו לא עולה, כי לא שמרנו דבר בזיכרון.

ניתן אגב לאחסן מידע בזיכרון, ולשחרר אותו בכל פעם שיש אינדיקציה שהזיכרון קטן ממשמעותית.

- שמירת מידע בזיכרון ותחזוקו לאורך ביצוע מתודות ה *map* אינו תואם את הירוח הפונקציונאלית.
של *map-reduce*.

- בשיטת הקומבינר יש overhead. הסביבה צריכה להפעיל שלב נוסף, לקרוא את הזוגות מהדיסק המ מקומי, לבצע לוקאלית *sort & shuffle*, וכו'.

- הקומבינר הוא רק אופטימיזציה, כלומר לא בהכרח יתרחש.

ניתן, אגב, לשלב בין השניים:

- השימוש בזיכרון יסכם את כל המופעים של מפתח אחד בספליט נתון לזוג אחד של V-K
הקומבינר ימזג את מופעי המפתח הנתון (מספרם כמספר הספליטים) וממזג אותם לזוג פלט אחד שי יצא מהמחשב הנתון

4.4 ספירת סדרות

נתמוך בספירה של זוגות מילים

ניתן להרחיב את התוכנית למספר מילים בודדות למספר זוגות מילים:

Class Mapper

Method Map(linId, line)

For w_1 in line

For w_2 in context(w_1) // the words around w_1

Emit(< w_1, w_2 >, 1)

Class Reducer

Method Reduce(pair, counts)

sum := 0

for (count in counts)

sum := sum + count

Emit(pair, sum)

גישה זו עשויה לייצר פלט לא 'קומפקטי' (של גוף לולאה אחד במתודת ה map):

<ילד אסור, 1>

<ילד מותר, 1>

<ילד טוב, 1>

<ילד אסור, 1>

<ילד טוב, 1>

<ילד טוב, 1>

<ילד רע, 1>

ניתן לארגן פלט זה באופן אחר:

<ילד, {אסור:2, מותר:1, טוב:3, רע:1}>

Class Mapper

Method Map(linId, line)

For w_1 in line

H := new Table

For w_2 in context(w_1)

```

H{w2} := H{w2} +1
Emit(w1,H)

```

Class Reducer

```

Method Reduce(w, Hs)
  Hsum := new Table
  for (H in Hs)
    add(Hsum,H)
  Emit(w, Hsum)

```

בשיטת אחת ('pairs'), המפתח הוא זוג מילים, והערך הוא מספר המופיעים של זוג המילים.
בשיטת השנייה ('stripes'), המפתח הוא מילה בודדת, והערך הוא טבלה עם מספר מופעי המילים השכנות.

איזה שיטה עדיפה?

נבחן שאלה זו במספר היבטים:

1. מספר המפתחות השונים (וכן מספר ה-a-k) הנוצרים ע"י המאפר

$O(n^2)$ Pairs: כמספר זוגות המילים
 $O(n)$ Stripes: כמספר המילים הבודדות

□ בגישהPairs צריך לשלווה יותר a-k, צריך למין יותר, וכו'

2. כמות המידע הנוצר

בגישה pairs הייצוג לא קומפקטי (כפי שראינו), בפרט יש חזרה שוב ושוב על המילה הראשונה.
גישה stripes אמנים יותר קומפקטיבית, אך נדרש סראליזציה מורכבת יותר.

☒ נראה שבגישהPairs נדרש להעביר בראשת הרבה יותר מידע.

3. סקלibility

ב pairs אין שמירה של מידע בזיכרון
ב stripes יש שמירה בזיכרון של טבלה קטנטנה המכילה את המילים השכנות למילה הראשונה, באותה שורה. זנית. ב-Reducer נדרש למזג את כל הטבלאות כך שזה עשוי להגיע ל- $O(n^2)$

4. אפקטיביות הקיבוץ המקומי

באייזו שיטה, ביצוע קיבוץ מקומי (ע"י קומביינר, או שמיירת מצב בזיכרון) תהיה אפקטיבית יותר? מילהבודדת חוזרת על עצמה הרבה יותר פעמים, מאשר זוג מילים חוזר על עצמו. כך שקיבוץ מקומי של מפתח המבוסס על מילהבודדת (כמו ב-stripes) יצמצם הרבה יותר את מספר ה-v-k שיוצאים מחשב בודד, מאשר קיבוץ מקומי של מפתח המבוסס על שתי מילים (כמו ב-pairs).

בדיקות ניסויית: ניסוי של LIN 2008

LIN השווה את שתי השיטות, עבור ספירה של זוגות מילים במאגר של כ-5.2 מיליון מסמכים (19 מחשבים, כל אחד עם שני מעבדים)

1. מספר ה-v-k הנוצרים ע"י המאפר

Pairs: נוצרו 2.6 מיליארד v-k

Stripes: נוצרו 653 מיליון v-k

2. גודל המידע הנוצר ע"י המאפר (ונשלח לרדיויסר)

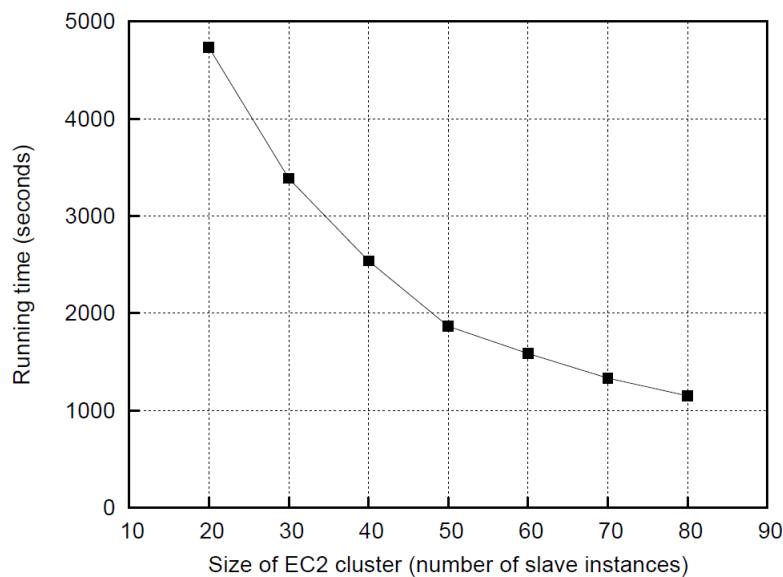
Pairs: GB32

Stripes: GB48 [מזהר, כי המילה הראשונה לא חוזרת על עצמה. LIN: בגלל סראליזציה גרוועה]

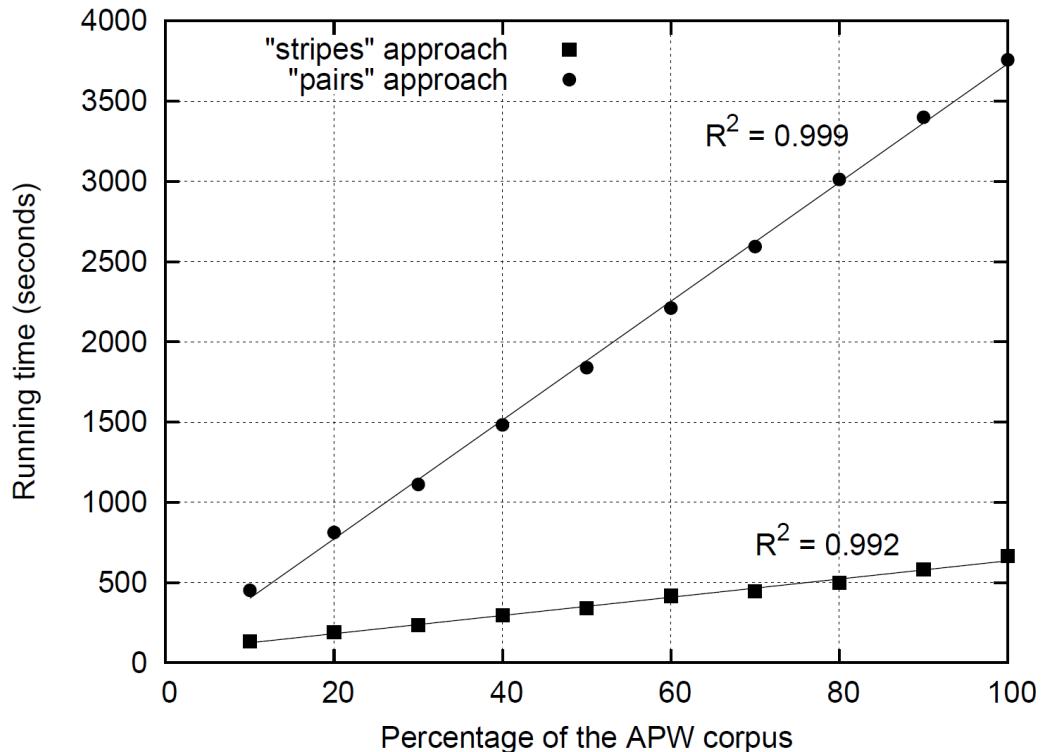
3. סקלibility.

לא היה בעיה.

o התגלה יחס הפוך בין מספר המחשבים לבין הריצה



ו. התגלה יחס ישיר בין גודל הקורפוס לזמן הריצה



4. אפקטיביות השימוש בקומביינר
 Pairs: 2.6 מיליארד □ 1.1 מיליארד
 Stripes: 29 מיליון □ 653

זמן הריצה הכללי:

Pairs: 62 min

Stripes: 11 min

מוחזרת התוצאה:

מדוע נוצר יותר מידע בstripes? לין: בגל הסריאליזציה

כיצד יתכן שהשיטה שיצרה יותר מידע רצח הרבה יותר? צוואר הבקבוק תמיד הוא התקשרות!

- לין: יש אופטימיזציה, בה נשליך ה-v-k לרדיווסר תוך כדי עבודה המאפייר
- יתכן (?) שליחות כמות גדולה של מידע תחת מספר קטן של שליחות, עיליה יותר משליחת של כמות קטנה יותר של מידע תחת מספר רב של שליחות.
- יתכן שבניטוי הספציפי, כל המחשבים ישבו על אותו מדף, כך שהתקשרות לא הייתה רלוונטית.

4.5 חישוב ההסתברות (חלוקת המונה במכנה)

זכור, אנו מעוניינים ליחס את נוסחת מודל השפה:

$$P_{MLE}(w_1 \dots w_n) = c(w_1 \dots w_n) / N$$

נתמך לשם פשוטות בסדרות של שתי מילים:

$$P_{MLE}(w_1 w_2) = c(w_1 w_2) / N$$

נתמך בוויאציה בה נדרש להסתברות המותנית: מה ההסתברות ש w_2 יופיע אחרי w_1

$$P_{MLE}(w_2 | w_1) = c(w_1 w_2) / \sum_j c(w_1 w_j)$$

בגישה stripes נסחה זו פשוטה למדי, כי למtodoת הרדיוס **מתקובל כל המידע הנדרש לחישוב הנסחה**, שהרי הסבiba מרכזת למtodoת ה `reduce` את כל המידע הקשור למפתח – במקרה שלנו את כל המידע הקשור למילה בודדת, ובפרט את מספרי המופעים של כל המילים שהופיעו אחריה.

Class Mapper

```
Method Map(lineId, line)
  For w1 in line
    H := new Table
    For w2 in context(w1)
      H{w2} := H{w2} + 1
    Emit(w1, H)
```

Class Reducer

```
Method Reduce(w1, Hs)
  // 1. Merge counts
  Hsum := new Table
  for (H in Hs)
    add(Hsum, H)
  // 2. Calculate the den
  den := 0
  for (<w2, count> in Hsum)
    den := den + count
```

```
// 3. Calculate the probabilities
for (<w2,count> in Hsum)
    Emit(<w1,w2>, count / den)
```

עבור המפתח 'ילד', נניח כי אחרי המיזוג (שלב 1) מתקבל: {אstor:4, מותר:7, טוב:9, רע:1} = Hsum

чисוב המכנה (שלב 2): den = 21

чисוב ההסתברויות (שלב 3):

'ילד' אטור 1/21, 'ילד' מותר 7/21, 'ילד' טוב 9/21, 'ילד' רע 1/21

בגישה pairs לעומת זאת, **המידע המתקבל במתודת הרדיוס הינו חלק עבור הנוסחה**, מתייחס למפתח שהוא זוג מילים אחד בלבד, כך שלא ניתן לחשב את הנוסחה:

Class Mapper

```
Method Map(lineld, line)
    For w1 in line
        For w2 in context(w1)
            Emit(<w1,w2>,1)
```

Class Reducer

```
Method Reduce(pair, counts)
    // 1. Merge counts
    sum := 0
    for (count in counts)
        sum := sum + count
    ???
```

אנחנו יודעים כמה פעמים הופיע w_1 עם w_2 (המונה בנוסחת ההסתברות), אך לא כמה פעמים הופיע w_1 עם מילים אחרות (המכנה).

פתרונות A [לא לנסוט בבית!]: נשמר את המונחים של כל הזוגות בזיכרון, ונחשב את ההסתברויות רק בסוף המשימה, על בסיס המידע שנאגר בזיכרון.

ו נdag לך שכל הזוגות המתחלים באונה מילה יגיעו לאותו Reducer על ידי שימוש **partitioner** המבוסס על המילה הראשונה בזוג.

[return key.first.hashCode() % iReducers]

ו נשמר בטבלה בזיכרון את כל הזוגות
ו נחשב את ההסתברויות בסוף

Class Reducer

Method Initialize()

H := new Table

Method Reduce(pair, counts)

// 1. Merge counts

sum := 0

for (count in counts)

 sum := sum + count

H{pair.first}{pair.second} := count

Method Close()

for (<w1,H2> in H)

//2. Calculate den

den := 0

for (<w2,count> in H2)

 den := den + count

// 3. Calculate the probabilities

for (<w2,count> in H2)

 Emit(<w1,w2>, count / den)

חיסרון: שמירה בזיכרון של כל זוגות המילים שהופיעו במאגר (ונוטבו לרדיוסר הנוכחי, N^2O). בעית סקליבליות.

פתרונות: שימוש במילון לשם מצומם המידע הנשמר בזיכרון

נமמש את מתודת ה compare של המחלקה **Pair** (המפתח), כך שכל הזוגות הפותחים באותה מילה יופיעו ברכף.

במקרה לדוגמה מילון בו זוגות המילים אינן מופיעות ברכף (כך שנדרש לשמר הכל בזיכרון):

}

ילד אסור: [2,2],

cosa yrock: [2,5],

ילד מותר: [3,4],

ילד טוב: [5,4],

cosa reu: [1],

ילד רע: [1]

}

נקבל במיון החדש רשיימה בה כל הזוגות הפתוחים באוטה מילה מופיעים ברצף:

```

}
ילד אסור: [2,2],
ילד טוב: [5,4],
ילד מותר: [3,4],
ילד רע: [1]
cosa יroke: [2,5],
cosa רוע: [1]
{
}
```

正如 מתחלפת המילה הראשונה, אנחנו יכולים להיות בטוחים שלא יופיעו בהמשך זוגות הפתוחים במילה הקודמת, כך שניתן כבר לחשב את ההסתברויות של הזוגות שפתחו במילה הקודמת, ולשחרר את הזיכרון.

Class Reducer

```

Method Initialize()
    H := new Table
    lastW1 := null

Method Reduce(pair, counts)
    // 1. Merge counts
    sum := 0
    for (count in counts)
        sum := sum + count
    if (pair.first <> lastW1) // the first word was changed
        //2. Calculate den
        den := 0
        for (<w2,count> in H)
            den := den + count
        // 3. Calculate the probabilities
        for (<w2,count> in H)
            Emit(<lastW1,w2>, count / den)
        H.clear()
        lastW1 = pair.first
    H{pair.second} := sum
```

```

Method Close()
    //2. Calculate den (for the last w1)
    den := 0
    for (<w2,count> in H)
        den := den + count

    // 3. Calculate the probabilities
    for (<w2,count> in H)
        Emit(<w1,w2>, count / den)

```

בגישה זו, נדרש לשמר בזיכרון רק את זוגות המילים הפותחות במילה מסוימת אחת (N(O))

פתרון ג: אי שימוש בזיכרון, בעזרת טכניות של שכפול מידע והיפוך סדר.

הרענון הכללי: נdag לך, שמתודת ה reduce תקבל קודם את המכנה של זוגות מילים הפותחות באותה מילה, ומיד אחר את הזוגות האלה, בזאה אחר זה.

```
{
   ILD * : [2,2,3,4,5,4,1],
   ILD אסור: [2,2],
   ILD טוב: [5,4],
   ILD מותר: [3,4],
   ILD רע: [1],
   cosa *, [2,5,1],
   cosa יroke: [2,5],
   cosa רועע: [1]
}
```

באופן זה, מתודת RIDIOS מקבלת מילה בודדת תחשב את המכנה, בעוד שמתודת RIDIOS מקבלת זוג מילים מחשבת מיד את ההסתברות שלהן ע"פ המכנה הנוכחי.

מימוש:

- המאפר ייצור גם מונימ עבור מילים בודדות (שכפול מידע, עבור כל זוג מילים שני - k)
- המין יעדיף תמיד * על פני כל מילה אחרת (היפוך סדר)

Class Mapper

Method Map(lineId, line)

For w_1 in line

```

For w2 in context(w1)
  Emit(<w1,w2>,1)
  Emit(<w1,*>,1)

Class Reducer
  Method Initialize()
    den := 0

  Method Reduce(pair, counts)
    // 1. Merge counts
    sum := 0
    for (count in counts)
      sum := sum + count
    if (pair.second = '*') /// den info
      den := sum
    else /// pair info
      Emit(pair, sum / den)

```

יתרונות: אין שימוש בזיכרון
חסרונות: הכפלת התקשרות

נוסף: מילוי משתני

דוגמא: נתונים חישנים רבים בעיר שיקאגו. כל חישון מייצר מידע כל פרק זמן קצר.

```

s1 t1 data1
s2 t1 data17
s1 t4 data80
s17 t3 data5
....

```

נניח כי אנו מעוניינים לבחון/לעבד את כל מה שקרה לכל חישון על ציר הזמן:

```

Class Mapper
  method Map (s,<t,d>)
    Emit(s,<t,d>)

```

```

Class Reducer
  Method Reduce(s,[<t,d>])

```

```
.... // process data
```

חיסרון: הסביבה צריכה ממיינת את המפתחות, אך לא את רשימת הערכים המתקבלת במתודת ה
reduce (במערכת של גול אגב היא ממויינת).

פתרונות א: נמיין בעצמנו בזיכרון.

Class Reducer

```
Method Reduce(s,[<t,d>])
  sort([<t,d>])
  .... // process data
```

חיסרון: לא סקלבייל

פתרונות ב: נמיין בדיאט...

פתרונות ג: נגרום לסביבה למין, על ידי העברת קרייטריון המיוון מהערך למפתח (value2key)

Class Mapper

```
method Map (s,<t,d>)
  Emit(>s,t,d)
```

Class Reducer

```
Method Reduce(<s,t>,[d])
```

....

באופן זה, סדר הפעלת מתודת הרדיוס יהיה:

```
reduce(<s1,t1>,[d1])
reduce(<s1,t2>,[d2])
reduce(<s1,t3>,[d3])
...
reduce(<s1,t786>,[d786])
reduce(<s2,t1>,[d943])
reduce(<s2,t2>,[d12])
...
```

כך שנוצר לנוסח את אלגוריתם העיבוד כמתבסס על הפעולות קודמות של מתודת ה reduce (עם
הנתונים עברו החישון הנוכחי בנקודות הזמן שקדמו)

חישון: הגדרה מסיבית של מספר המפתחות השונים, מספר החישונים למספר החישונים בכל נקודת זמן.

Join .5

5.1 מבוא

פעולת ה join צריכה, משדقت נתונים מקורות שונים על פי קритריונים מוגדרים.
לדוגמא, בסיס נתונים רלוונטי:

Table Users

Id, name, phone

Table Actions

Id, userId, type, date,...

ניתן בעזרת פעולה Join לשדר נתונים לקו עם פרטי הפעולות שלו:

```
Select users.name, users.phone, actions.date, actions.type  
From users inner join actions on users.id = actions.userId
```

בפרק זה, נבחן כיצד ניתן לבצע שידור נתונים שכזה, כאשר הנתונים אינם מאוחסנים בסיס נתונים רלוונטי אלא מפוזרים בקבצים שונים של Big-Data.

דוגמא:

Users1.txt

7 danny

9 yossi

Users2.txt

7 44598977

9 14305523

Actions1.txt

1 7 click 1/1/18

2 9 call 9/9/88

Actions2.txt

3 9 browse 16/3/19

4 7 call 8/5/20

ברצוננו לכתוב תוכנית map-reduce המחזירה את פרטי כל לקוח (שם, מספר טלפון) עם פרטי כל אחת המפעולות אותן ביצע (סוג, תאריך):

danny 445989 1/1/18 click
danny 445989 8/5/20 call
yossi 143055 9/9/88 call
yossi 143055 16/3/19 browse

5.2 Join של שתי טבלאות'

Mapper-Side Join .1

אבחן: האופי של שתי הטבלאות (קבצי users וקבצי actions) שונות:

- אחת הטבלאות (users) קטנה יותר באופן משמעותית מהטבלה השנייה (actions).
- קצב הגידול של הטבלה הקטנה איטי הרבה יותר מקצב הגידול של הטבלה הגדולה.

נניח כי ניתן לאחסן את כל נתונים הטבלה הקטנה בזיכרון.

עיצוב התוכנית:

- כל אחד מהמאפרים יטען לזכרון בשלב ההתחל שלו את כל נתונים הטבלה הקטנה (כלומר, יקרא את השורות בקבצי users ממערכת הקבצים המבוזרת, ויתען אותן לטבלה בזיכרון).
- לדבר זה יש מחיר של תקשורת, וחשש לביעית סקליביליות.
- הקלט לג'וב יהיה קבצי הטבלה הגדולה.
 - משימת המאפר מקבלת ספליט של אחד הקבצים של הטבלה הגדולה (אחד מקבצי actions)
 - מתודת ה map מקבלת תיאור של רשומה אחת בטבלה הגדולה (תיאור של פעולה אחת)
 - מתודת ה map מבצעת שידור של הרשימה הנтונה מהטבלה הגדולה עם הרשימה המתאימה של הטבלה הקטנה המואחסנת בזיכרון (שידור של פעולה אחת עם פרטי לקוח המאוחסנים בזיכרון, ע"פ מספר הלוקום).
- בגישה זו השידור נעשה כבר בשלב ה Map, כך שאין צורך כלל ב Reduce.

Method Initialize

```
T1 := new Table
table1dir = conf.get("table_1_dir")
for file in table1dir
    for line in file
        record1 := parse(line)
        T1{record1.getJoinKey()} = record1
```

Method Map(id2, record2)

```
id1 := record2.getForeignKey()
record1 := T1{id1}
Emit(id1, crossProduct(record1, record2))
```

לשם בהירות, נכתב שוב את הקוד באופן ספציפי עבור המקרה של Users, Actions

Class Mapper

Method Initialize

```
TUsers := new Table
usersDir = conf.get("table_users_dir")
for usersFile in usersDir
    for line in file
        user := parse(line)
        TUsers{user.getJoinKey()} := user //user.getJoinKey() implemented as returning userId
```

Method Map(actionId, action)

```
userId := action.getJoinKey() // implemented as returning userId field of Action class
user := TUsers{userId}
Emit(userId, crossProduct(user,action))
```

הפרודצורה crossProduct ממומשת בהתאם לשדות שנבחרו להציג ב'Select'. בדוגמה שלנו היא תמומש כמחזירה את הריבועית user.name, user.phone, action.type, action.date

יתרונ: חיסכון בתקשורת – אין צורך לשלוח את נתוני הטבלה מהמابر לרדיוסר

חסרון: תקשורת בהעברת קבוע הפעלה הקטנה לכל המאפרים. הנחה על הזיכרון.

Reducer-Side Join .2

אם לא ניתן להניח שאפשר לאחסן בזיכרון את נתוני הפעלה הקטנה.

חלק מהמאפרים יקבלו ספליט מהטבלה הראשונה, וחלק מהמאפרים יקבלו ספליט של הטבלה השנייה. כלומר, מתודת ה map עשויה לקבל לעיתים פרטיו לקוח, ולעתים פרטיו פועלה.

Mapper1 – Users1.txt

Mapper2 – Actions1.txt

Mapper3 – Actions2.txt

Mapper4 – Users2.txt

הרעיון הכללי: מתודת ה map תעביר את המידע שהוא קיבל כערך, כאשר המפתח הוא ה foreign key (כלומר, בין אם היא קיבלה פרטיו לקוח, ובין אם היא קיבלה פרטי פעולה, הם ישלחו תחת המפתח של מספר הלקווח). באופן זה, מתודת הרדיוס תקבל עבור מספר לקוח נתון, את כל פרטי המידע הרלוונטיים עבורו (פרטיו הלקווח וכל הפעולות שהוא ביצע) כך שהוא יוכל לשדר ביניהם.

בעיה: כדי לבצע את השידור, יש להבחן במקור של כל ערך בראשימת הערכים שהגיעו למетодת הרדיוס.

לדוגמא:

Mapper1 – Users1

<7, <7, danny>>

<9, <9, yossi>>

□

<7, <7,danny>>

<9, <9,Yossi>>

Mapper2 – Actions1

<1, <1 7 click 1/1/18>>

<2, <2 9 call 9/9/88>>

□

<7, <1,7, click 1/1/18>>

<9, <2,9 call, 9/9/88>>

Mapper3 - Actions2.txt

<3, <3 9 browse 16/3/19>>

<4, <4 7 call 8/5/20>>

□

<9, <3,9, browse, 16/3/19>>

<7, <4,7, call, 8/5/20>>

Mapper4 - Users2.txt

7 445989

9 143055

□

<7,<7, 445989>>

<9,<9, 143055>>

Reducer1:

<7,[<7,danny>,<4,7,call,8/5/20>,<7,445989>,<1,7,click 1/1/18>

]>

יש לשדר את "danny 445989" (פרט המשתמש) עם "call, 8/5/20" (פרט פעולה אחת).
אחר כך יש לשדר את "danny 445989" (פרט המשתמש) עם "click 1/1/18" (פרט הפעולה
השנייה).

אך איננו יודעים, מאיין בא כל ערך.

Reducer2:

<9,[<9,Yossi>,<3,9,browse,16/3/19>,<9,143055>,<2,9,call,9/9/88>>

יש לשדר את "Yossi 143055" (פרט המשתמש) עם "call, 9/9/88" (פרט פעולה אחת).
אחר כך יש לשדר את "Yossi 143055" (פרט המשתמש) עם "browse, 16/3/19" (פרט
הפעולה השנייה).

אך איננו יודעים, מאיין בא כל ערך.

□ נסיף לכל ערך 'טג זיהוי', ונסוג את הערכים בזיכרון ע"פ טג זה במתודת הרדיו לפני השידור.

Class Mapper

Method Initialize

tag := pickTag(split.name)

Method Map(key, record)

Emit(record.getJoinKey(), <**tag**,record>)

Method Reduce(id, **taggedRacords**)

// classify values to 2 groups (table1, table2)

H := new Table

for taggedRecord in taggedRecords

H{taggedRecord.tag} □ taggedRecord.record

// cross product of the data for the given id of the two tables

crossProduct(H)

```
<7, [<'u',<7,danny>>,<'a', <4, call, 8/5/20>>, <'u',<7, 445989>,<'a', <1, click 1/1/18>>]>
```

```
H: { 'u' : [danny, 445989], 'a' : [1 click 1/1/18, 4 call 8/5/20] }
```

יתרון:

- אין צורך להניח שהטבלה הראשונה נכנסת לזיכרון
- אין צורך להעביר את קבצי הטבלה הקטנה לכל המאפרים.

חיסרון:

- יש יותר תקשורת, שליחת זוגות מה Mappers ל Reducers (מספר כל הרשומות בטבלאות)
- בעיית סקלביות, אנו מניחים שניין לאחסן בזיכרון את כל הפרטים של מפתח בודד (את פרטי הליקוי וכל הפעולות אותן ביצע מעולם)

.3 Join Reduce-Side Join עם מין משני

הרעיון הכללי: נמיין את רשימת הערכים המגיעה למוגדות הרדיוס, כך שקדם יופיעו ברכף כל הערכים מהטבלה הקטנה, ורק אח"כ הערכים בטבלה הגדולה.

```
<7, [<'u',<7,danny>>,<'u', <7, 445989>>,<'a' , <4, call, 8/5/20>>,<'a',< 1, click 1/1/18>>]
```

באופן זה, נדרש לאחסן בזיכרון רק את הנתונים מהטבלה הקטנה עבור מפתח אחד.

```
H: [danny, 445989]
```

כיצד נמיין ביג-דאטה? בטכניקת key-to-value, כלומר על ידי העברת קרטיטריון (התג) למפתח.

```
{
  '>u',7>: [danny, 445989[
    '>a',7>: ]<4, call, 8/5/20>,<1, click 1/1/18>[
      '>u',9>: [yossi, 143055 [
        '>a',9>: ]< 3, browse, 16/3/19>,<2, call, 9/9/88>[
      ]
    ]
  ]
}
```

Class Mapper

Method Initialize

```
  tag := pickTag(split.name)
```

Method Map(key, record)

```
  Emit(<tag,record.getJoinKey(),record)
```

Class Reduce(taggedId, values)

Method Initialize

```
lastId := -1  
record1 := null
```

Method Reduce (**taggedId**, records)

```
If (taggedId.id != lastId) // table one (users)  
    record1 = records  
Else // table two (actions)  
    For record2 in records  
        crossProduct(record1, record2)  
    lastId = taggedId.id
```

יתרונות: שימוש מינימלי בזיכרון (רק נתונים הטללה הקטנה עברו מפתח אחד)

חיסרונות: פ' שניים זוגות (אותה כמות מידע)

5.3 Join של שלוש טבלאות

דוגמא: נתונים סטודנטים באוניברסיטה

Students

```
17 danny  
248 yossi
```

Courses

```
925 spl  
10 os
```

StudentsCourses

```
17 925 90  
248 10 80
```

ברצוננו לשדר נתונים משלושת הטעיות:

Danny, spl, 90
Yossi, os, 80

אפשרויות ראשונה: נבצע שני סיבובים של שידור שתי טבלאות.

- שידוך הطالאות StudentsCourses | Students studId:

danny 925 90

yossi 10 80

- שידוך הטבלה Courses עם הטבלה שנוצרה בשלב הקודם, בעזרת המפתח הזר courseId:coursesId

Danny, spl, 90

Yossi, os, 80

אפשרות שנייה: נסדר את כל שלוש הطالאות בסיבוב אחד של Map-Reduce

בעה מרכזית: שידוך שלושת הطالאות מבוסס על שני מפתחות זרים שונים.

כלומר,

מافר שמקבל נתונים של סטודנט (ספליט של Students), אינו יכול 'לשלו' אותם לקורס המתאים, כי הוא לא מקבל את מספרי הקורסים שהסטודנט לומד.

מافר שמקבל נתונים של קורס (ספליט של Courses), אינו יכול לשלו' אותם לסטודנט המתאים, כי הוא לא מקבל את מספרי הסטודנטים שלוקחים את הקורס.

[מافר שמקבל ציון של סטודנט בקורס (ספליט של CoursesStudents) יכול לשלו' לסטודנט ו/או לקורס המתאים]

□

מافר שמקבל נתונים סטודנט ישלח אותם לכל הקורסים (גם לכלה שהסטודנט לא לומד)

- מافר שמקבל נתונים קורס ישלח אותם לכל הסטודנטים (גם לכלה שלא לומדים את הקורס)

בעיה: כיצד נדע מה ה- id של כל הסטודנטים והקורסים?

☒ נמפה אותם בעזרת פונקציית hash ייחודית, בתחום $1 \dots M/N$, כאשר M/N הוא המספר הנתון של הסטודנטים/הקורסים.

לדוגמא:

נגדיר פונקציית האש 1 הממפה מספר סטודנט לתחום 1..2:

$\text{h1}(248) = 1$

$\text{h1}(17) = 2$

נגדיר פונקציית האש 2 הממפה מספר קורס לתחום 1..2:

$\text{h2}(925) = 1$

$\text{h2}(10) = 2$

יעזוב:

בاهינתן שלוש טבלאות T,S,R עם קשיי היחס הבאים (כל שתיים מהן קשורות על ידי מפתח זר משותף):

R)A,B)

Students(studName, studId)

S(B,C,E)

StrudentsCourses (studId, courseId, grade)

T(C,D)

Courses(courseId, courseName)

כאשר נתון כי מספר ערכי C,B (=מספר הרשומות בטבלאות R,T) הם M,N בהתאם (ידוע כמו סטודנטים וקורסים קיימים), ונתנות פונקציות האש הממפה את הערכים באופן ייחודי לתחומים אלו (h1,h2).

- מפתחות: נגדיר כל מפתח כזוג <j,i>, כאשר j מייצג אינדקס של מפתח מהטבלה R (אינדקס של סטודנט), ו j מייצג אינדקס של מפתח מהטבלה T (אינדקס של קורס).
בדוגמא שלנו:

<1,1> : yossi,spl

<2,1> : danny, spl

<1,2> : yossi, os

<2,2> : danny, os

- נגדיר את פעולות המאפרים באופן הבא:

o המאפר שמקבל רשומה מהטבלה b,a,<

לדוגמא: Students < studName,studId > מ

'שולח' (כלומר emit) את הערך ,<a> תחת המפתחות <h1(b),y> עברו כל y בתחום

M...1

<17,danny>

?

<2,1>,<17,danny>

<2,2>,<17, danny>

<248,yossi>

?

<1,1>,<248,yossi>
<1,2>,<248,Yossi>

ו המאפר שמקבל רשומה מהטבלה S: <b,c,e>

לדוגמא: StudentsCourses מ < studentId, courseId, grade >

שולח את הערך <b,c,e> תחת המפתחות <h1(b),h2(c)>

<17,925,90>

¶

<2,1>,<17,925,90>

<248,10,80>

¶

<1,2>,<248,10,80>

ו המאפר שמקבל רשומה מהטבלה C: <c,d>

לדוגמא: Courses מ < courseId,courseName >

שולח את הערך <d,c> תחת המפתחות <h2(c),h1(a)> עבור כל x בתחום 1...N

<925,spl>

¶

<1,1>,<925,spl>

<2,1>,<925,spl>

<10,os>

¶

<1,2>,<10,os>

<2,2>,<10,os>

באופן זה, הרדיוסר יקבל תחת מפתח המיצג שידור בין הנתונים בטבלאות, את כל המידע הרלבנטי לשידור זה. במידה והתקבל מידע/ערכים רק משתי טבלאות (סטודנט שלא לומד קורס), אין שידור.

בדוגמא שלם, הזוגות שהמאפרים יצרו הם:

```

<2,1>,<17,danny>
<2,2>,<17, danny>
<1,1>,<248,yossi>
<1,2>,<248,Yossi>
<2,1>,<17,925,90>
<1,2>,<248,10,80>
<1,1>,<925,spl>
<2,1>,<925,spl>
<1,2>,<10,os>
<2,2>,<10,os>

```

לאחר ה shuffle & sort תתקבל ברדיוסר הטבלה הבאה:

```

} <1,1> : [<248,yossi>, <925,spl>],
  <1,2> : [<248,Yossi>, <248, 10, 80>, <10,os>],
  <2,1> : [<17,danny>, <17,925,90>, <925,spl>],
  <2,2> : [<17, danny>, <10,os>] }

reduce(<1,1>, [<248,yossi>, <925,spl>]) \ 
reduce(<1,2>, [<248,Yossi>, <248, 10, 80>, <10,os>]) \ "Yossi,os,80"
reduce(<2,1>,[<17,danny>, <17,925,90>, <925,spl>]) \ "Danny,spl,90"
reduce(<2,2>,[<17, danny>, <10,os>]) \

```

הגדרת מרחב המפתחות כפרטט:

ביצוב הנוכחי, קיימ מפתח שונה לכל קומבינציה של שני המפתחות הזרים (לכל קומבינציה של סטודנט-קורס).

נגיד החלטה זו כפרטט: ניתן לקבוע מראש את מספר המפתחות השונים. במידה ומספר המפתחות קטן ממספר הקומבינציות האפשריות בין המפתחות הזרים (מספר המפתחות קטן ממספר הסטודנטים X מספר הקורסים), פונקציית האש תמפה כמה קומבינציות למפתח אחד.

הגדרת מרחב המפתחות כפרטט, מאפשר מצד אחד לקבוע כמה רמת המקובלות המקסימלית (משימת רדיוסר מוגדרת לפחות למפתח אחד, אך שמספר ממשימות הרדיוסר הרצות במקביל הן לכל היותר מספר המפתחות) ביחס למשאבי החישוב המוצאים, ומצד שני משפיעת על סיבוכיות ניהולם (יש לארגן ולמיין את מופעי המפתחות השונים ברדיוסר).

עיצוב:

- נגיד את פונקציית האש כך שהן ימכו כמה מפתחות לאוטו אינדקסו.
- נגיד את מספר המפתחות: $n \cdot l = k$

- המאפר שמקבל רשומה מהטבלה b,a,<R> ישלח את הערך b תחת המפתחות <y,(b)> עברו כל y בתחום 1...n
- המאפר שמקבל רשומה מהטבלה d,c,<T> ישלח את הערך d תחת המפתחות <c,(d)> עברו כל x בתחום 1...n

עבור הדוגמא שלנו:

$$n \cdot k = 1 \cdot 2 = 2$$

פונקציות האש夷 עשוות למפות שני סטודנטים לאותו אינדקס, או שני קורסים לאותו אינדקס:

$$h1(17) = 1$$

$$h1(248) = 1$$

$$h2(10) = 1$$

$$h2(925) = 2$$

```
} <1,1> : [<17,danny>, <248, 10, 80>, <10,os>, <248,yossi>],  
<1,2> : [<17, danny>,<248,Yossi>, <925,spl>,<17,925,90>] }
```

```
reduce(<1,1>, [<17,danny>, <248, 10, 80>, <10,os>, <248,yossi>]) ↳ "Yossi,os,80" -reduce(<1,2>,  
<17, danny>,<248,Yossi>, <925,spl>,<17,925,90>) ↳ "Danny,spl,90"
```

חומר של שלוש טבלאות ציקליות (מעגליות), עם שלושה מפתחות זרים

במקרה זה גם הטבלה השלישייה קשורה לראשונה, באופן הבא:

$$R(A,B,E)$$

$$S(B,C,F)$$

$$T(C,A,G)$$

כלומר כר אחית מהטבלאות קשורה לשתי הטבלאות האחרות.

נדרש לשדר נTONIM משלוש הטבלאות.

נתון, כי אנו מעוניינים במרחב מפתחות בגודל K.

נגיד את מידת השכפול של המידע הנשלח משלושת הטבלאות (כלומר, כמה פעמים תישלח כל

רשומה לכל אחת מהטבלאות), ע"י הגדרת ח,מ,א כאשר $h \cdot m \cdot n = k$

מרחב המפתחות יהיה שלשות $\{z,y,x\}$ כאשר:

$$x = 1 \dots l$$

$$y = 1 \dots m$$

$$z = 1 \dots n$$

נגיד שלוש פונקציות hash המפות למפתחות מהטבלאות השונות למרחב האינדקסים שלו:

$$h1(a) = 1 \dots l$$

$h2(b) = 1 \dots m$

$h3(c) = 1 \dots n$

- כל מאפר המקבל רשומה מטבלה אחת, ישלח את המידע למפתח עם האינדקס של טבלה זו, והאינדקסים של הטבלאות האחרות.

המאפר של הטבלה הראשונה E(A,B,C)

מקבל רשומה $\langle e, b, a \rangle$, ושולח אותה תחת המפתחות $\langle z, h1(a), h2(b) \rangle$, עברו כל $z = 1 \dots z$

המאפר של הטבלה השנייה F(B,C,D)

מקבל רשומה $\langle f, c, d \rangle$, ושולח אותה תחת המפתחות $\langle x, h2(b), h3(c) \rangle$, עברו כל $x = 1 \dots x$

המאפר של הטבלה השלישית G(C,A,D)

מקבל רשומה $\langle g, a, c \rangle$, ושולח אותה תחת המפתחות $\langle y, h1(a), h3(c) \rangle$, עברו כל $y = 1 \dots y$

שאלות

- כיצד ניתן לקבוע את מידת השכפול של שליחת כל רשומה בכל טבלה?
- ע"פ האלגוריתם, כל רשומה בטבלה הראשונה נשלחת ח פעמים, כל רשומה בטבלה השנייה נשלחת 1 פעמים, כל רשומה בטבלה השלישית נשלחת 2 פעמים. מהם הערכים האופטימליים של a, b, c , בהתאם לגודל ואופי הטבלאות.
- איזו מהышיות, שני סיבובים של 2-way (M-R) או סיבוב אחד של שידור משולש (3-way), עדיפה? באילו תנאים?

נבחן תחילה מה מינימום פעולה התקשרות הנדרשות בגישה השנייה של 3-way, ותחת אילו ערכיהם של a, b, c .

מספר פעולה התקשרות בין ה Mapper ל Reducer באלגוריתם האחרון של השידור המשולש הוא:

- שליחת כל רשומה בטבלה a R פעמים
- שליחת כל רשומה בטבלה b S פעמים
- שליחת כל רשומה בטבלה c T פעמים

כלומר: $m \cdot t + l \cdot s + k \cdot r$, כאשר t, s, r הם מספר הרשומות בטבלאות T, S, R בהתאם.

כיצד נבחר את a, b, c באופן אופטימאלי כך שייהי כמה שפחות פעולות תקשורת?

קיימת טכניקה בשם Lagrange Multiplier למציאת מינימום לוקאלי של ביטוי מהצורה:

$$f(x, y, z) - \lambda \cdot g(x, y, z)$$

נתאר את מספר פעולות התקשרות מהמאפרים לרדיוסרים (ב 3-way) כביטוי מהצורה הזאת:

$$f(l, m, n) = r \cdot t + l \cdot s + k \cdot m$$

נדיר את הפונקציה g באופן מלאכותי, כדי לקבל את מספר פגולות התקשרות בצורה לוגrangle:

$$g(l,m,n) = l \cdot m \cdot n - k (=0)$$

הfonקציה $h, m, a(f)$ מייצגת את מספר פעולות התקשרות מה Mappers ל Reducers.

הfonקציה $h, m, a(g)$ שווה תמיד ל-0.

כך שהביטוי $h, m, a(f) - h, m, a(g) \cdot \lambda$, בדומה הלאגרז'אית, מייצג את מספר פעולות התקשרות.

טכנית Lagrange Multiplier תמציא את הערכים האופטימליים של הפרמטרים h, m, a שייתנו ערך מינימלי לביטוי, כלומר מספר מינימלי של פעולות התקשרות.

שימוש הטכנית:

- נגדיר את הביטוי כשווא ל-0

$$f(l,m,n) - \lambda \cdot g(l,m,n) = 0$$

$$r \cdot n + s \cdot l + t \cdot m - \lambda \cdot (l \cdot m \cdot n - k) = 0$$

- נגזר את המשוואה על כל אחד מהמשתנים

газירה על l :

$$s - \lambda \cdot m \cdot n = 0 \quad \square \quad s = \lambda \cdot m \cdot n$$

газירה על m :

$$t - \lambda \cdot n = 0 \quad \square \quad t = \lambda \cdot n$$

газירה על n :

$$r - \lambda \cdot l = 0 \quad \square \quad r = \lambda \cdot l$$

- נכפול כל משוואה במשתנה הגזירה ונקבל:

$$l \cdot s = \lambda \cdot m \cdot n \cdot l = \lambda \cdot k$$

$$m \cdot t = \lambda \cdot n \cdot l = \lambda \cdot k$$

$$n \cdot r = l \cdot n \cdot m = \lambda \cdot k$$

משוואות אלו מגדירות את הערכים האופטימליים עבור h, m, a כך שהביטוי המקורי (כלומר מספר פעולות התקשרות) יהיה מינימלי.

נשחק קצת עם המשוואות כדי לקבל את הנתונים שימושיים אותן:

נכפיל את שלוש המשוואות ונקבל:

$$l \cdot s \cdot m \cdot t \cdot n \cdot r = \lambda^3 k^3$$

$$\lambda = \sqrt[3]{\frac{rst}{k^2}}$$

נציב ונקבל:

$$l = \sqrt[3]{\frac{krt}{s^2}}$$

$$m = \sqrt[3]{\frac{krs}{t^2}}$$

$$n = \sqrt[3]{\frac{kst}{r^2}}$$

הצבת ch, m, t , האופטימליים בביטוי המקורי ($ch \cdot t + ch \cdot s + ch \cdot z$) ייתן את מספר פעולות התקשורת המינימלי בין Mapper לReducer:

$$O(3\sqrt[3]{krst})$$

[משוואות אלו מגדירות גם את היחס בין גודל הטבלאות t, s, z לבין הערכים האופטימליים של ch, m, t .]

לדוגמא: $t/s = ch/z$

כדי להחליט איזו שיטה (2-way or 3-way) עדיפה, נבדוק מה ה'סיבוכיות' כל שיטה, כולם מה פועלות תקשורת מתבצעות בכל שיטה.

אילו פועלות תקשורת מבוצעות בתוכנית R-M?

- o העברת הקלט מהקברים ל Mappers
- o העברת הפלט של Mappers ל Reducers
- o [העברת הפלט לספריית הפלט]

נבחן את מספר פעולות אילו בשתי השיטות:

way-2

- o קריית הקלט למאפר

סיבוב ראשון: $s + z$ [קריית כל רשומה בכל אחת משתי הטבלאות הראשונות R | S]
 סיבוב שני: $k \cdot s + t$, כאשר k הוא הסתבותות השידור של רשומה בטבלה z ורשומה בטבלה s
 [קריית כל רשומה מהטבלה השלישית Z, וכל רשומה מפלט הסיבוב הראשון, כולם משידורי הרשומות של הטבלאות R | S]

- o העברת פלט המאפר לרדיוסר
כל מה שהמאפר מקבל הוא מעביר לרדיוסר
- סיבוב ראשון: $s + r$
- סיבוב שני: $k \cdot s + r + t$

מספר פעולות התקשרות הוא: $2(t + r + s)$

way-3

- o קריית הקלט למאפר
 - + $r + s + t$ [קריית כל אחת מהרשומות בשלושת הטבלאות T,S,R]
 - o העברת הפלט של המאפר לדיזור
- ראינו קודם, כי בבחירה אופטימאלית של t, r, s , מספר פעולות התקשרות הוא בסדר גודל של

$$3\sqrt[3]{krst}$$

כל עוד:

$$2(rsp + r + s + t) > r + s + t + 3\sqrt[3]{krst}$$

עדיף 3 way

במילים אחרות, אם הטבלאות הדוקות, כלומר הסתברות גבוהה לשידור ביניהן (ק גובה), עדיף 3-way: הרבה פלט לקריאה מהט'bob הראשון ב 2-way, מעט מידע שנשלח סתם ב 3-way. אחרת, עדיף 2-way.

לסיכום: ההחלטה האם עדיף 2-way או 3-way תלויות במידת ההבדוקות בין שתי הטבלאות, והחלטה על כמה שכפול המידע הנשלח מכל טבלה תלויות בגודל הטבלאות השונות.

Fuzzy Join 6.4

הבעיה: נתונה קבוצה של אובייקטים ('טבלה' אחת עם רשומות) S, ופונקציה f המתקבלת כפרמטר שני אובייקטים.

נדרש להפעיל את הפונקציה על כל שני אובייקטים בקבוצה/בטבלה.
[כלומר, מדובר בשידור fuzzy של רשומות מאותה טבלה]

לדוגמא: קבוצה של כל המילים במילון השפה העברית, ופונקציה המתקבלת שתי מילים ומחזירה ציון עד כמה המילים דומות. ברצוננו ליצור פלט בו מופיע עבור כל זוג מילים מידת הדמיון ביניהן.

בעיצוב סדרתי התוכנית פשוטה:

```
for obj1 in S
    for obj2 in S
        if (obj1 != obj2)
            f(obj1, obj2)
```

אם מדובר בbig-data, נדרש לעיצוב של תוכנית R-M.

כיצד נוצרים לכך שמתודות הרדיווס תקבל כל פעם שני אובייקטים (שוניים)?

☒ נשתמש באותה טכניקה של הגדרת מרחב המפתחות:

- נגדיר את מרחב המפתחות כזוגות $\langle y, x \rangle$ כאשר x מציין את האינדקס של האובייקט הראשון ו- y מציין את האינדקס של האובייקט השני (להפעלה ע"י f)
- מתודת `map` תשלוח כל אובייקט שהוא מקבלת לכל האובייקטים האחרים, כלומר תחת מפתחות שביהם האינדקס שלו מופיע עם כל אינדקס אחר.
- כדי להבטיח שזוג אובייקטים ימופה לאותו מפתח נאלץ את המפתחות $\langle y, x \rangle$ כך שהערך הקטן מופיע משמאל: $y < x$.

Class Mapper

```
method Map(id1, obj)
for (id2 := 1 to |S|)
    if (id1 > id2)
        Emit(<id1,id2>, obj)
    if (id2 > id1)
        Emit(<id2,id1>, obj)
```

Class Reducer

```
method reduce(<x,y>, [obj1, obj2])
f(obj1,obj2)
```

דוגמא:

- 1ILD
- 2TOB
- 3ILDHA
- 4TOWBA

Mapper
 $\langle 1, <1, <1, \text{ILD} \rangle \rangle$
 $\langle 2, <1, <2, \text{ILD} \rangle \rangle$
 $\langle 2, <1, <3, \text{ILD} \rangle \rangle$
 $\langle 2, <1, <4, \text{ILD} \rangle \rangle$
 $\langle 2, <2, \text{TOB} \rangle \rangle$
 $\langle 3, <1, <2, \text{TOB} \rangle \rangle$
 $\langle 3, <2, <3, \text{TOB} \rangle \rangle$
 $\langle 3, <2, <4, \text{TOB} \rangle \rangle$

<3,ילדה>

[?]

<1,3> ילדה

<2,3> ילדה

<3,4> ילדה

<4,טובה>

[?]

<1,4> טובה

<2,4> טובה

<3,4> טובה

Reducer

[ילד, טוב]<>[<1,2

[ילד, ילדה]<>[<1,3>

[ילד, טובה]<>[<1,4>

[טוב, ילדה]<>[<2,3>

[טוב, טובה]<>[<2,4>

[ילדה, טובה]<>[<3,4>

הערות:

- אם המפתחות של האובייקטים אינם רציפים, נמפה אותם למרחב רציף של אינדקסים ייחודיים עם פונקציית hash כמו קודם.
- אם נדרש, ניתן לצמצם את מספר המפתחות השונים ל-K, על ידי מיפוי של כמה מפתחות לאותו אינדקס כמו קודם (כגון שבמתודות הרדיוס, יש לשדר יותר מזוג מילימטרים)

סיבוכיות:

תקשורת

קריאת הקלט: |S|, מספר האובייקטים בקבוצה

העברת פלט המאפר לרדיאס: |S-1| · |S|

בעיות למידה

הזכרנו בתחילת הקורס, כי קיימות דרכים שונות ללמוד מערכת מוחשבת שפה אנושית (רלבנטי לכל בעיה אחרת):

- לימוד הסתברותי מול לימוד מונחה חוקים
- לימוד מונחה מול לימוד שאינו מונחה

ובנוסף:

- מודל גנרטיבי מול מודל דיסקרימינטיבי

בחלק זה של הקורס נבחן מספר בעיות בתחום של לימוד שפה, באופן הסתברותי, בלימוד מונחה ובלימוד שאינו מונחה. הקלט בבעיות אלו יהיה בדרך מאגר גדול של נתונים טקסט.

- ניתוח צורני
 - [הסתברותי, לא מונחה, גנרטיבי - מודל מרקוב]
- ניתוח תחבירי
 - [הסתברותי, מונחה, דיסקרימינטיבי - בניית מסותג]
- סיווג מסמכים ומידול נושאים
 - o Clustering [לא מונחה]
 - o LSA [לא מונחה]
 - o Topic Modeling o
- ■ A [לא מונחה]
- ■ רשותות נירונית [מונחה]

6. ניתוח צורני / תיוג חלקו דבר / מציאת הקטגוריה הלקסיונאלית של המילים

6.1 מבוא

דוגמה: ספר עזר לרופא Shinnyim בהוצאה כתר

ניתן לפרש את המשפט בשני אופנים שונים. יש לפרש אותו נכונה עברו כל אפליקציה שעוסקת בשפה. אבחנה: כדי למצוא את הפירוש הנוכחי, מספיק למצוא/להכיר מהי הקטגוריה הלקסיונאלית (פועל, שם עצם, תואר) של המילים. לדוגמה, מספיק לגלוות האם עזר הוא פועל (עוזר) או תואר (עוזר)
 ☒ מציאת הקטגוריה הלקסיונאלית של המילים במשפט, הינה רכיב חיוני ובסיסי בכל מערכת של ניתוח טקסט.

הבעיה

נתון: משפט בשפה טבעית
יש למצוא מהי **הקטגוריה** של כל אחת מהמילות במשפט.

מהן הקטגוריות הלексיקליות?

לא כל כרך פשוט. נתמקד בקטגוריות ברורות יחסית:

פועל [הלך, אכטוב, תלמודו, יגיע...]
שם עצם [כסא, שולחן, ילדים, ילדי, מכתבים, רחובות...]
תואר השם [עצב, שמחה, גבויים...]
תואר הפועל [לאט, מהר...]
מילות יחס [מעל, מתחת, לפני, אחרי,...]
מילות קשר [או, לאחר,...]

ההחלטה לחלק את המילים בשפה לקטגוריות אלו, קשורה לתכונות הצורניות המשותפות למילים בכל קבוצה:

פועל: משנה את הצורה שלו ע"פ מאפיינים של岷, כמות, גוף, זמן [הלה, הלה, הלה, הלכת, הלכת, הולר, אלר]

שם עצם: משנה את הצורה שלו ע"פ מאפיינים של כמות וסמיכות, ניתן לידעו [ילד, ילדים, ילדי, הילד]

במקרים רבים, קיים כבר משאב של מילון/לקסיקון המספק רשיימה של קטגוריות אפשריות לכל מילה:
ילד: פועל, שם עצם
את: שם עצם (את), מילת יחס (את), מילת גוף (את)
נתקדם בבעיה פשוטה יותר, של בחירת הקטgorיה המתאימה מבין הקטgorיות במילון עבור מילה נתונה,
ע"פ ההקשר שבו היא נאמרה.

לדוגמא:

יש לי את חפירה – שם עצם
ראיתי את הילד – מילת יחס
את באה לסרט? – מילת גוף

נבחן **מודל הסתברותי** המאפשר ללמידה באופן שאינו מונחתי (אך ורק ע"פ קלט של טקסט פשוט): **מודל מרקוב חבוי (Hidden Markov Model)**, מודל גנרטיבי, ונממש את אלגוריתם הלמידה שלו בתבנית Map-Reduce:

- נלמד על מודל מרקוב לכשעצמנו
- נישם את המודל עבור הבעיה שלנו (מציאת הקטgorיה של כל מילה במשפט)
- נעצב מחדש את אלגוריתם הלמידה במודל בתבנית Map-Reduce

6.2 מודל מרקוב

נתונה תופעה מסוימת בעולם. אנחנו מעוניינים להסביר אותה.

לדוגמא: צבע הדגל על סוכת המציג בחוף משתנה מיום ליום. מדוע הצבע משתנה? אלו גורמים בעולם משפיעים על צבע הדגל? כיצד הם משפיעים? כיצד ניתן לנבأ את הצבעמחר?

כדי לענות על שאלת זו:

- נגידר **מודל**: נניח הנחות מסוימות שייעזרו לנו להסביר את התופעות.

לדוגמא: 1) הצבע של הדגל תלוי בעוצמת הרוח; 2) עוצמת הרוח יכולה להיות {חלש, בינוני, חזק}; 3) עוצמת הרוח משפיעה על הדגל שבאותו יום, ועל עוצמת הרוח ביום המחרת.

- אלגוריתם הלמידה ילמד את הפרמטרים של המודל לאור נתוני העבר. לדוגמא: נלמד מצביעי הדגלים בעבר כיצד משפיעה עוצמת הרוח על צבע הדגל, כיצד משפיעה עוצמת הרוח היום על עוצמת הרוחמחר. כיצד?

מודל מורכב מספק מסגרת לבניית מודל שכזה, ולימוד אוטומטי של הפרמטרים שלו לאור תוצאות עבר.

הגדרת המודל

מודל מורכב מוגדר על ידי השלישי μ : S, K :

S אוסף מצבים [הגורמים המשפיעים על התופעות, בדוגמא שלנו עוצמת הרוח: {חלש, בינוני, חזק}]

K אלףית פלט [קבוצת כל התופעות שברצוננו להסביר, בדוגמא שלנו צבעי הדגל: {לבן, אדום, שחור}]

μ מודל הסתברותי, קובע את היחס בין המצבים [הגורמים] לבין הפלט [תופעות] באמצעות שתי מטריצות ויקטור:

$A_{i,j} = \{A\}$ מגדרה את ההסתברות לעבר מצב i במצב j
[מה ההסתברות שתיה רוח חזקה אחרי רוח שלישי]

$B_{i,k} = \{B\}$ מגדרה את ההסתברות לפט k כאשר המצב הוא i
[מה ההסתברות לדגל אדום כאשר הרוח חזקה]

$\Pi_j = \{\Pi_j\}$ מגדר את ההסתברות להתחיל במצב j
[מה ההסתברות שביום הראשון תהיה רוח חלש]

דוגמא:

$S = \{\text{חלש, בינוני, חזק}\}$

$K = \{\text{לבן, אדום, שחור}\}$

μ :

A

חזק	בינוני	חלש	
0.2	0.5	0.3	חלש
0.3	0.3	0.4	בינוני
0.4	0.1	0.5	חזק

B

שחור	אדום	לבן	
0.1	0.2	0.7	חלש
0.3	0.4	0.3	בינוני
0.7	0.2	0.1	חזק

$$(\Pi = (0.3, 0.4, 0.3)$$

תהליך מרקובי

אנו רואים במודל מרקלוב מכונה, העוברת ממצב ל המצב, ומיצרת אגב כך פלט, באופן הסתברותי.
כלומר, היא פועלת ע"פ הסכמה הבאה:

$t := 1$

Start in state s_{t1} according to Π

Forever do

Emit observation symbol k according current state and according to B

Move from state s_{t1} to state s_{t2} according to A

$t := t + 1$

מודל שכזה מכונה 'מודל גנרטיבי', או 'מודל יוצר', שכן התופעות שאוותן אנו רוצים להסביר הן למעשה הפלט של המכונה ההסתברותית המבוססת על הגורמים.

נשים לב, כי האופן שבו שבעה המכונה אינו ידוע, איננו יודעים דרך אילו מצבים היא עברה, אנחנו רואים רק את הפלט שלה. לכן הוא מכונה 'מודל מרקלוב חbowי', המצביעים שחוללו את הפלט הנוכחי חbowים.

שלוש בעיות יסוד

1. מה ההסתברות של סדרת תופעות

(שחור לבן לבן) P

$P(o_1 \dots o_T)$

2. בהינתן סדרת תופעות, מהם הגורמים (סדרת המצביעים) המסבירים כי טוב תופעות אלו.

$$P(X_1 X_2 X_3 | \text{לבן})$$

$$\operatorname{argmax} X_1 X_2 X_3$$

כלומר, מה ההצעה האופטימלית של X_3, X_2, X_1 (לערבים חזק בינווי וחלש) שתיתן את הערך הגבוה ביותר להסתברות. כלומר, שטסביר הכי טוב את התוצאות לבן שחור.

ובמקרה הכללי:

$$P(X_1 \dots X_T | o_1 \dots o_T)$$

$$\operatorname{argmax} X_1 \dots X_T$$

3. בהינתן מאגר של סדרות תופעות ('קורפוס'), מהו המודל ההסתברותי המתאים ביותר עבורו.

$$P(O | \mu)$$

$$\operatorname{argmax} \mu$$

בדוגמא שלנו, מהו המודל ההסתברותי הטוב ביותר שמסביר את צבעי הדגלים ביום השני לאחריות. כלומר, מהם הערכים בהתאם לטבלאות A, B וווקטור Π הנוגאים את הערך הגבוה ביותר עבור הפונקציה: $P(\mu | \text{flag colors in 100 years})$

נבחן בעיות אלו:

1. חישוב הסתברות פלט נתון

$$P(\text{לבן} | \text{לבן})$$

מה ההסתברות שהיו דגלי לבן שחור?

תלויה ביעילות הרוח. נבחן את האפשרויות השונות ונסכם את ההסתברות לפלט (לבן שחור) עבור כל אחת מהן:

o **חלש חלש חלש**

$$\Pi_{\text{שחור, חלש, חלש, אלבן, חלש, b}} = b_{\text{חלש, חלש, אלבן, חלש, b}}$$

o **חלש חלש בינווי**

$$\Pi_{\text{שחור, בינווי, b, אלבן, חלש, b}} = b_{\text{בינווי, חלש, אלבן, חלש, b}}$$

o **חלש חלש חזק**

$$\Pi_{\text{שחור, חזק, חלש, אלבן, חלש, b}} = b_{\text{חזק, חלש, אלבן, חלש, b}}$$

o **בינווי חלש חלש**

$$\Pi_{\text{שחור, חלש, חלש, אלבן, חלש, b}} = b_{\text{חולש, חלש, אלבן, בינווי, b}}$$

...

o חזק חזק חזק

$$\pi_{x_1 \dots x_T} b_{x_1, o_1} a_{x_2, o_2} \dots a_{x_{T-1}, T} b_{x_T, o_T}$$

ובמקרה הכללי:

$$P(o_1 \dots o_T) = \sum_{x_1 \dots x_T} \pi_{x_1} b_{x_1, o_1} a_{x_2, o_2} \dots a_{x_{T-1}, T} b_{x_T, o_T}$$

מספר האפשרויות אקספוננציאלי (בגודל סדרת המצבים), לא ישם חישובית.
□ נשתמש בטכנית פשוטה של תכונות דינמי. כלומר, נזכיר הסתברויות שכבר חישבנו ונשתמש בהן.

נגידר שני מבני נתונים (טבלאות): α, β

הטבלה α תגדיר את ההסתברויות להגעה בתהיל' המركבי למצב i בזמן t עבור סדרת פלט נתונה-הנתונה. כלומר הערך בתא i, t מציין את ההסתברות להגעה למצב i בזמן t , נסמן: $\alpha_i(t)$
 ניתן להגדיר אינדוקטיבית את ערכי התאים האלו באופן הבא:

$$\begin{aligned}\alpha_i(1) &= \pi_i b_{i, o_1} \\ \alpha_i(t) &= \sum_j \alpha_j(t-1) a_{j, i} b_{i, o_t}\end{aligned}$$

לדוגמא, עבור המודל שלנו, וסדרת הפלט (לבן לבן שחור):

$$\begin{aligned}\alpha_1(1) &= \pi_1 b_{1, 0.3} = 0.7 \times 0.3 = 0.21 \\ \alpha_1(2) &= \pi_1 b_{1, 0.4} = 0.7 \times 0.4 = 0.28 \\ \alpha_1(3) &= \pi_1 b_{1, 0.1} = 0.7 \times 0.1 = 0.07\end{aligned}$$

$$\begin{aligned}\alpha_2(1) &= \pi_2 b_{2, 0.3} + \alpha_1(1) a_{1, 2} b_{2, 0.3} = 0.3 \times 0.3 + 0.21 \times 0.2 = 0.183 \\ \dots\end{aligned}$$

前进式计算 Forward

הטבלה β תגדיר את ההסתברויות ליצר את שאר הפלט, כאשר בזמן t אנחנו במצב i . כלומר הערך בתא i, t מציין את ההסתברות לצאת ממצב i בזמן t ולסימם את סדרת הפלט, נסמן: $\beta_i(t)$
 ניתן להגדיר אינדוקטיבית את ערכי התאים האלו באופן הבא:

$$\begin{aligned}\beta_i(T) &= 1 \\ \beta_i(t) &= \sum_j a_{i, j} b_{j, o_{t+1}} \beta_j(t+1)\end{aligned}$$

לדוגמא, עבור המודל שלנו, וסדרת הפלט (לבן לבן שחור):

$$\beta_1 = (3)_{\text{חולש}}$$

$$\begin{aligned} B_1 &= (3)_{\text{בינוני}} \\ \beta_1 &= (3)_{\text{חזק}} \end{aligned}$$

$$\begin{aligned} \beta_2 &= (3)_{\text{חולש}} + (3)_{\text{בינוני}} b_{\text{חולש}, \text{חולש}} a \\ &\dots \end{aligned}$$

前进方向的计算称为 Forward 后退方向的计算称为 Backward

כדי לחשב את ההסתברות של סדרת פלט נתונה (זו הבעה שבה אנחנו עוסקים): $P(o_1 \dots o_T)$
 נבחר נקודת זמן כלשהי t (לא משנה איזה, הכל יצא אותו דבר). בנקודת זמן זו ייצקן כי נהייה
 במצבים שונים – נסcom את האפשרויות להגעה לכל מצב בזמן זה, ולהמשיך משם עד הסוף (עד
 זמן T).

$$P(o_1 \dots o_T) = \sum_i \alpha_i(t) \beta_i(t), \text{ for some } 1 \leq t \leq T$$

$$P(o_1 \dots o_T) = P(\text{לבן לבן שחור}) + P(\text{לבן לבן שחור}) + P(\text{לבן לבן שחור})$$

cut the probabilities if it is calculated the probabilities β, α , namely $D^2 | S | O$.

b. מציאת הסבר טוב לפלט

בהינתן סדרת תופעות, מהם הגורמים (סדרת המצבים) המסבירים הכי טוב תופעות אלו.

$$\begin{aligned} P(X_1 X_2 X_3 | \text{לבן לבן שחור}) \\ \text{argmax } X_1 X_2 X_3 \end{aligned}$$

כמובן, מה הצבה האופטימלית של X_1, X_2, X_3 (לערך $\{\text{חזק}, \text{בינוני}, \text{חולש}\}$) שתיתן את הערך הגבוה
 ביותר להסתברות. כמובן, שתסביר הכי טוב את התופעות לבן לבן שחור.

ובמקרה הכללי:

$$\begin{aligned} P(X_1 \dots X_T | o_1 \dots o_T) \\ \text{argmax } X_1 \dots X_T \end{aligned}$$

כדי למצוא את סדרת המצבים האופטימלית, נחשב את ההסתברות עבור כל סדרת מצבים אפשרית, ונבחר את זאת שנותנת את ההסתברות הכי גבוהה.

בדוגמה שלנו, נחשב את ההסתברויות (על פי המודל, a וכל זה):

(חלש חלש|שחור לבן לבן)
 (בינוי חלש חלש|שחור לבן לבן)
 (חזק חלש חלש|שחור לבן לבן)
 (חלש חלש בינוי|שחור לבן לבן)
 ...
 (חזק חזק חזק|שחור לבן לבן)

בעיה: מספר האפשרויות אקספוננציאלי (במספר המ מצבים)
פתרון: גם כאן, נستخدم בתכנות דינמי. כלומר, נשמר בזיכרון הסתברויות שחישבנו, ועל פיה נחשב את ההסתברויות הבאות.

בדומה לאלפא וביתא בבעיה א, נגדיר טבלאות אחרות למדעה ופי:
 $\delta_i(t)$ מגדיר את ההסתברות של הטובה ביותר להגעה בזמן t למצב i
 $\psi_i(t)$ מגדיר את המצב שקדם לו בסדרת המצביעים הטובה ביותר להגעה בזמן t למצב i

אלגוריתם ויטרבי:

Initialization

$$\delta_i(1) = \pi_i b_{i,o_1} \quad (6.1)$$

Induction

$$\delta_i(t) = \max_j \delta_j(t-1) a_{j,i} b_{i,o_t} \quad (6.2)$$

$$\psi_i(t) = \arg \max_j \delta_j(t-1) a_{j,i} b_{i,o_t} \quad (6.3)$$

Termination and path readout

$$\hat{X}_T = \arg \max_i \delta_i(T) \quad (6.4)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \max_i \delta_i(T) \quad (6.5)$$

ג. אימון המודל

בاهינתן מاجر של סדרות תופעות ('קורפוס'), מהו המודל ההסתברותי המתאים ביותר?

$$\underset{\mu}{\operatorname{argmax}} \quad P(\mathcal{O} | \mu)$$

ובמילים אחרות, כיצד 'לאמן' את המודל לאור כמות גדולה של תופעות שנצפו בעבר (לאור פלט גדול של 'המכונה').

נשים לב, כי אם המידע במاجر היה כולל לא רק את הפלט/התופעות (צבי הדוגלים) אלא גם את 'הסביר' שלהם, יכולומר את המצב/עוצמת הרוח הכל יומ (....,<אדם, חזק>,<לבן, בינוני>) חישוב ההסתברויות עבור התאים במטריצות היה פשוט.

$a_{i,j}$ = number of transitions from state i to state j / number of transitions from state i

$b_{i,k}$ = number of emitions of output-symbol k from state i / number of transitions of state i

π_i = number of times a sequence starts from state i / number of sequences

כינינו לימוד שצהה, על בסיס מاجر שבו יש גם הסבר לתופעות, לימוד מונחה.
אנו,自然, מעוניינים במשימה מתאגרת יותר של לימוד לא מונחה – המاجر לאימון המודל כולל רק את התופעות ללא ההסבירים (רק את צבי הדוגלים בדוגמה שלנו).

לשם כך קיימת משפחה של אלגוריתמים המכונה EM (Expectation-Maximization).

המבנה הכללי של אלגוריתמים אלו:

- o התחלה ממודל כלשהו (כמו אתחול אקראי של הפרמטרים של המודל או מאיתחול מושכל יותר)
- o כל עוד המודל מתעדכן משמעותית
 - קריית מاجر הנתונים ופרשנותו לאור המודל הנוכחי (Expectation)
 - עדכון המודל כך שיסביר טוב יותר את הנתונים (Maximization)

עבור מודל מרקוב, קיימים אלגוריתם EM, המכונה אלגוריתם באום-וילש, והמוגדר באופן הבא:

$a_{i,j}$ = number of **expected (according to current model)** transitions from state i to state j / number of **expected (according to current model)** transitions from state i

$b_{i,k}$ = number of **expected (according to current model)** emitions of symbol k from state i / number of **expected (according to current model)** visits in state i

π_i = number of **expected (according to current model)** times a sequence starts from state i / number of sequences

Expectation

$$\begin{aligned}\alpha_i(1) &= \pi_i b_{i,o_1} \\ \alpha_i(t) &= b_{i,o_t} \sum_j \alpha_j(t-1) a_{j,i}\end{aligned}\tag{6.6}$$

$$\begin{aligned}\beta_i(T) &= 1 \\ \beta_i(t) &= \sum_j a_{i,j} b_{j,o_{t+1}} \beta_j(t+1)\end{aligned}\tag{6.7}$$

Maximization

$$\hat{\pi}_i = \frac{\alpha_i(1) \beta_i(1)}{\sum_j \alpha_j(1) \beta_j(1)}\tag{6.8}$$

$$\hat{a}_{i,j} = \frac{\sum_{t=2}^T \alpha_i(t-1) a_{i,j} b_{j,o_t} \beta_j(t)}{\sum_{t=1}^{T-1} \alpha_i(t) \beta_i(t)}\tag{6.9}$$

$$\hat{b}_{i,k} = \frac{\sum_{t:o_t=k} \alpha_i(t) \beta_i(t)}{\sum_{t=1}^T \alpha_i(t) \beta_i(t)}\tag{6.10}$$

משפט באום-אייגע (1967): $P(O|\mu) = P(\mu|O)$
במילים אחרות, בכל סיבוב של קריית המאגר ועדכון המודל, מקבלים מודל לא פחות טוב מהקדום. כלומר, האלגוריתם מתכנס.

6.3 מודל מרקוב עבור בעיית תיוג חלקி הדיבור / מציאת הקטגוריה הלוקסיקאלית של כל מילה במשפט

זכור, בהינתן משפט, אנו מעוניינים למצוא את הקטגוריה (שם עצם, פועל, תואר...) של כל מילה. כדי ללמד זאת באופן אוטומטי ממאגר טקסט לא מנומתח, בחרנו במודל מרקוב.

כיצד ניתן לייצג במודל מרקוב את הבעיה שלנו?

נחשב על המודל כמכונה שמייצרת מילים. כלומר כל סדרת פלט היא משפט.

מהם המצביעים במכונה זו? כאמור, מהם הגורמים היוצרים את המילויים (התופעות)?
 הקטגוריות הלוקסיקליות.

- o המכונה בוחרת קטgorיה התחלתית למשפט, על פי הווקטור ד [שם עצם]
- o מכאן ואילך, עד סוף המשפט
 - המכונה מייצרת מילה ע"פ הקטגוריה הנוכחית ומטריצה B [ילך]
 - המכונה עוברת לקטגוריה חדשה [פועל]

דוגמא למודל שכזה:
S {שם עצם, פועל, תואר}
K {ספר, עזר}

: מ

A

תואר	פועל	שם עצם	
0.4	0.5	0.1	שם עצם
0.2	0.1	0.7	פועל
0.1	0.5	0.4	תואר

B

עזרה	מספר	
0.1	0.9	שם עצם
0.4	0.6	פועל
1	0	תואר

P
 $(0.6, 0.2, 0.2)$

כיצד נבחר את ההסתברויות בטבלאות?

נאותחל את הטבלאות, ונאמן אותן לאור מאגר גדול של משפטים, עם אלגוריתם באומ-וולש.

בاهינתן משפט, כיצד נבחר את הקטגוריות המתאימות?

'ספר עזר'

- יש למצוא את סדרת המצבים (=קטגוריות) המסתברת ביוטר שיצרה פלט זה (=המשפט הנתון) –
אלגוריתם ויטרבי לעיל.

6.4 עיצוב אלגוריתם האימון בתבנית Map-Reduce

מסתבר, שאיכות אלגוריתם האימון במקרים רבים תלויות בגודל הקורפוס (מאגר הטקסטים).

- יש לתמוך בהרצת האלגוריתם על big data, מאגר עצום של טקסטים.
- יש לעצב את אלגוריתם האימון כתוכנית Map-Reduce

עיצוב

הנחות על הזיכרון: נניח כי ניתן לשמר את המודל ההסתברותי (Π, B, A) בזיכרון. כלומר ניתן להכיל בזיכרון $|S|^2 + |A| |S| + |S|$ מספרים.

משימת המאפר (קטgorית, טבעית): מקבל סדרות פלט, מבצע עליה Expectation, ומחשב את חלק ה Maximization האפשרי בקונטקט הקיימים - המונחים.

משימת הרדיוסר (קטgorית): מקבל המונחים מסדרות פלט שונות, כפי שהושבו ע"י הפעולות שונות של מתודת ה map, ומשקלו אותם לכדי השלמת ה maximization.

מהן ייחדות המידע עליו עובדות מתודות המפה וה `reduce`? ובמילים אחרות, מה המפתח במתודת המפה ומה המפתח במתודת השילמה (כך נוכל

נעצב, כמו תמיד, את ייחדות המידע של הפונקציות, כך שיהיו מינימליות עבור ביצוע הפעולה (כך נוכל לקבל את הפעולות יותר ע"פ רצוננו):

כדי לבצע expectation נדרש סדרת פלט אחת (משפט אחד לדוגמא) \square מתודת המפה מקבל סדרת פלט אחת.

כדי לבצע maximization נדרש כל התובנות עבור שורה אחת בטבלה (B, A או כל הווקטור Π) \square מתודת ה reduce מקבל את כל המידע עבור שורה אחת בטבלאות.

עיצוב האלגוריתם

Mapper

- טווען באתחול את המודל ההסתברותי האחרון (מהסיבוב הקודם)

- בהינתן סדרת פלט במתודת ה `map`, בונה מודל חדש (Π, B, A) היכול את המונחים של נוסחאות ה `maximization` לאור ה `expectation` של המשפט הבזק.
- שולח כל שורת מונחים במודל החדש שנוצר לרדיויסרים, תחת מפתח השורה.

Reducer

- מקבל במתודת ה `reduce` רשימת מופעים של אותה שורה (מאחר הטבלאות, או Π), כאשר כל מופיע מכיל את המונחים של כל תא על פי ה `maximization`
- מיזוג (ע"י סכימה) המונחים של כל תא (מיוזג התבוננות מכל סדרות הפלט במאגר)
- אבחנה: בכל נוסחאות ה `maximization`, המכנה הוא סכום המונחים בשורה \square חישוב המכנה של כל התאים, סכום המונחים של התאים בשורה.
- חישוב ההסתברות בכל תא במודל החדש, כחלוקת המונה שלו במכנה הנ"ל.

Class Mapper

Method Initialize

```
<S,A,B, Π>      := loadPrevModel
```

Method Map(seqId, seq)

```
// 1. calc expectation
alpha := forward(seq, A,B, Π)
beta := backward(seq, A,B, Π)
```

```
// 2. Create new Model
Π' := new Vector
A' := new Table
B' := new Table
```

```
// 3. Calc counters
for (t := 1 to |seq|)
    for s1 in S
        B'[s1,seqt] := B'[s1,seqt] + alpha[s1,t] · beta[s1,t]
        If (t=1)
            Π'[s1] := alpha[s1,1] · beta[s1,1]
        for s2 in S
            A[s1,s2]' := A[s1,s2]' + alpha[s1,t] · A[s1,s2] · B[s2,seqt+1] · beta[s2,t+1]
```

// 4. Send the counters to the reducers, line by line

```
Emit("pi",Π')
```

```
for s in S
```

```

Emit("A:" + s, A[s])
Emit("B:" + s, B[s])

```

Class Reducer

```

method Reduce(lineId, lineCounters)
    // 1. Merge line counters (as given by various output sequences)
    // [ [1,2,3] [4,5,6] ] ⊕ [5,7,9]
    newLine := new Vector
    for lineCounters in lineCounters
        add(newLine, lineCounters)

    // 2. Calculate den (as the sum of the line counters)
    // [5,7,9] ⊕ 21
    den = 0
    for counter in newLine
        den := den + counter

    // 3. Calc the probabilities of the line
    // [5,7,9] ⊕ [5/21, 7/21, 9/21]
    for i := 1 to | newLine |
        newLine [i] = newLine [i] / den

    // 4. Export result (for next training iteration)
    Emit(lineId, newline)

```

Class Combiner

```

method Reduce(lineId, lineCounters)
    mergedLine := new Vector
    for lineCounters in lineCounters
        add(mergedLine, lineCounters)
    Emit(lineId, mergedLine)

```

6.5 אתחול המודל

ראינו, כי אלגוריתם האימון מתחילה ממודל קלשו ומשפר אותו שוב ושוב.

כיצד נתחל את המודל?

1. אקראי
נזרק מספרים בתאים, וננرمל כר שכל שורה שווה ל-1.

A

תואר	פועל	שם עצם	
0.4	0.5	0.1	שם עצם
0.2	0.1	0.7	פועל
0.1	0.5	0.4	תואר

B

עדן	ספר	
0.1	0.9	שם עצם
0.4	0.6	פועל
1	0	תואר

$$\Pi \\ (0.2, 0.3, 0.5)$$

2. התפלגות אחידה

A

תואר	פועל	שם עצם	
0.333	0.333	0.333	שם עצם
0.333	0.333	0.333	פועל
0.333	0.333	0.333	תואר

עזרה	ספר	
0.5	0.5	שם עצם
0.5	0.5	פועל
0.5	0.5	תואר

$$\Pi \\ (0.333, 0.333, 0.333)$$

מסתבר, שאთחלים שכאלו נוטים להוביל את אלגוריתם האימון לנקודת מקסימום מקומית. כלומר, אלגוריתם האימון של מודל מרקוב רגש מאוד לתנאי ההתחלה (לאתחול של המודל).

[אימון המודל מנקודת התחילה זו, נותן בעברית מודל עם כ-80% דיוק]

□ ננסה לשפר את אתחול המטריצות. בפרט נתמך באתחול טוב יותר של מטריצה B.

3. אתחול מטריצה B ע"פ מיליון נתון

נתון מיליון/לקסיקון, המגדיר עבור כל מילה את רשימת הקטגוריות המתאימות למילה זו.
לדוגמא:

ספר: [שם עצם, פועל]

עזרה: [שם עצם, פועל, תואר]

עזרה	ספר	
1	1	שם עצם
1	1	פועל
1	0	תואר

לאחר נרמול:

עזרה	ספר	

0.5	0.5	שם עצם
0.5	0.5	פועל
1	0	תנוור

אימון המודל מנקודת התבילה זו, נותן בעברית מודל עם 87% דיוק.

ננסה לפתח שיטות אתחול טבות יותר.

4. אתחול מטריצה B על פי שכיחות המילים הדומות

דוגמא: המילה 'את'

המילון מצין 3 קטגוריות אפשריות למילה 'את':

ו שם עצם (את, את חפירה)

ו מילת יחס (את, ראייתי את יוסי')

ו מילת גוף (את, את באה לבקר')

אתחול ע"פ המילון ייתן הסתברות אחידה לשולש הקטגוריות, והסתברות 0 לכל השאר.

המילה 'את' לכשעצמה (ללא הקשר של משפט) מתפרקת ברוב המוחלט של המקרים כ밀ת יחס, במספר מקרים קטן כ밀ת גוף, ובמספר אפסי כשם עצם.

נמשקל את האפשרויות במילון, כך שכל קטgorיה תקבל הסתברות כללית לפגוש בטקסט במילה תחת קטgorיה זו.

ו שם עצם (את, כמו את חפירה) [0.01]

ו מילת יחס (את, כמו ראייתי את יוסי') [0.9]

ו מילת גוף (את, כמו את באה לבקר') [0.09]

אם מאגר הטקסטים היה מנוטח (כך שלא כל מילה מצינית הקטgorיה שלה), אז היינו פשוט סופרים כמה פעמים הופיעה המילה בכל קטgorיה ומחלקים.

כיצד נעשה זאת, כאשר המאגר אינו מנוטח?

אבחןנה: כל אחת מהקטgorיות, גוזרת באופן שונה מילים דומות ע"י הטיוות.

לדוגמא:

את שם עצם: את, אתים, האת, האטים, את... .

את כ밀ת גוף: את, אתה, אתם, אתן...

את כ밀ת יחס: את, [אותו, אותה, אותם, אותן, אתכם, אתכן, אותןן]

□

אם את היא שם עצם באופן כללי, אנו מוצפים לפגוש במאגר מילים כמו: את, אתים, האת, האתים, את*.*

אם את היא מילת גוף באופן כללי, אנו מוצפים לפגוש במאגר מילים כמו: את, אתה, אתם, את*.*
אם את היא מילת יחס באופן כללי, אנו מוצפים לפגוש במאגר מילים כמו: את.

□ נספור את מופעי המילים הדומות המוצפות בכל קטגוריה, ונחשב את ההסתברות של קטגוריה על פי השכיחות היחסית של המילים בכל קטגוריה.

אלגוריתם לויינגר-אורן-אייטי (1995) עושה זאת, עם כמה איטרציות אימון:

Input:

w – A word with k analyses: A_1, \dots, A_k .

SW_1, \dots, SW_k – The similar words sets of analyses A_1, \dots, A_k .

sw – A word in some SW set.

$C(sw)$ – The number of occurrences of sw in the training corpus.

$Inc(sw)$ – A set of indexes representing the analyses for which sw is a member in their SW set, i.e., $Inc(sw) = \{l : 1 \leq l \leq k, sw \in SW_l\}$

ε – A prespecified threshold indicating the convergence of the algorithm.

Internal Variables:

P_j^i – The approximated morpho-lexical probability of A_j in the i -th iteration.

$SumAnal_j$ – The sum over the contribution of all the words in SW_j .

$AvgAnal_j$ – The average contribution of a single word in SW_j to $SumAnal_j$.

The Algorithm:

```

 $P_1^0 := P_2^0 := \dots := P_k^0 := 1/k;$ 
 $i := 0;$ 
repeat
     $i := i + 1;$ 
    for  $j := 1$  to  $k$  do begin
         $SumAnal_j = \sum_{sw \in SW_j} C(sw) \times (P_j^{i-1} / \sum_{l \in Inc(sw)} P_l^{i-1});$ 
         $AvgAnal_j := SumAnal_j / size(SW_j)$ 
    end;
    for  $j := 1$  to  $k$  do
         $P_j^i := AvgAnal_j / (AvgAnal_1 + \dots + AvgAnal_k)$ 
until (  $\max_j |P_j^i - P_j^{i-1}| < \varepsilon$  ) .

```

אימון המודל לאחר אתחולו בשיטה זו נותן 91% דיוק בניתוח מורפולוגי מלא (כ-4000 קטגוריות) עבור טקסט עברי, ודיוק של 93% בתיאוג של הקטגוריות (כ-40 קטגוריות)

האם נדרש למש את האלגוריתם בתבנית Map-Reduce?

- האלגוריתם מבוסס על תשתיות של ספירת מילים בודדות: טבלה המציינת כמה פעמים מופיעה בקורסוס כל מילה אפשרית.

□ הרצת תוכנית WordCount ב-Map-Reduce מתחילה הקורס.

- האלגוריתם עצמו, עבר כל מילה במיילון, עובר בכל איטרציה על כל אחד מהנתונים שלא במיילון. יש כ-100,000 מילים במיילון.

לכל מילה יש ממוצע שלושה ניתוחים.

האלגוריתם מתכנס די מהר (אין הרבה איטרציות)

□ ניתן להרייך סדרתית

5. אתחול ע"פ הקשרי המילה

ניתן לזרות שני חסרון בגין הקודמת (הבסיסת על ספירת ה햇יות בכל קטgorיה):

- השיטה מתאימה בעיקר לשפות עם מורפולוגיה עשירה (כמו עברית וערבית), ככלומר, שפות בהן מילה משנה את צורתה בהתאם למאפייני הטיה שונים בכל קטgorיה, כמו מין, כמות, זמן ועוד. השיטה לא תהיה אפקטיבית בשפות עם מורפולוגיה עניה (כמו אנגלית), ככלומר שפות בהן המילה לא כל כך משנה את צורתה עם שינוי מאפייני מין, כמות ועוד.
- גם עבור שפות עם מורפולוגיה עשירה, לעיתים ה햇יות בקטגוריות השונות זהות, וכך שלא ניתן להסתמך על שכיחותן היחסית.

לדוגמה: המילה 'של'

של, מילת יחס: של, שלך, שלכם, שלנו (הטיות סיומת נומיניב)

של, שם עצם: של, שלך, שלכם, שלנו (הטיות סיומת שייכות)

נעיר את הסתברויות הקטגוריות השונות עבור מילה נתונה ע"פ הקשרים שבהם המילה מופיעה.

לדוגמה: המילה 'של'

מהם הקשרים שבהם המילה 'של' מופיעה?

בית של בובות

הספר של יוסי
מעריב של שבת
לבשתי של ירוק
מצאתי של יפה

ניתן לראות ש'הסבiba הטבעית' של המילה 'של' בשתי הקטגוריות שוניה. בשני הבטים:

ו מהן המילים משנה צידי המילה 'של'

מילות יחס:
בית _ בובות
הספר _ יוסי
מעריב _ שבת

שם עצם:
לבשתי _ ירוק
מצאתי _ יפה

ו מהן הקטגוריות משנה צידי המילה 'של'

מילות יחס:
שם-עצם _ שם-עצם
שם-עצם _ שם-פרטי
שם-פרטי _ שם-עצם

שם עצם:
פועל _ תואר
פועל _ תואר

□ נגידר את ההסתברות של כל קטgorיה למילה, ע"פ תכניות המילים מסביבה בקורסופים, וע"פ תכניות הקטגוריות סביבה בקורסופים (ביחס לתכניות עבור הקטגוריות האחרות למילה)

נגידר את ההסתברות של קטgorיה מסוימת c , עבור הקשר מילים מסוימים w

$$p(t|c) = \sum_w p(w|c) \cdot p(t|w)$$

(מילה יחס | בית _ בובות)
(מילה יחס | ראייתי _ ירוק)
(שם עצם | בית _ בובות)

(שם עצם | ראיתי __ יroke)k

...

כיצד נדע מה ההסתברות של הקטגוריה עברו כל אחת מהAMILIM?

$$p(t|w) = \sum_c p(c|w) \cdot p(t|c)$$

(AMILIT YCHOS | SHL)k

(שם עצם | SHL)k

...

בשתי הנוסחאות שהגדרכנו, יש ארבעה ביטויים: $w|c$, $p(c|w)$, $p(t|c)$, $p(w|t)$

נשים לב שהביטויים $w|c$, $p(c|w)$, $w|t$, $p(t|w)$ ניתנים לחישוב באופן פשוט (MLE) על פי מאגר של טקסט.

נשים לב גם, כי ניתן לאותחל את הביטוי $w|t$ על המילון בהתפלגות איחידה (גישה ג) או על מילון משקל המילון לפי שכיחות הטעיות (גישה ד)

□

o אתחול

- נחשב את $w|c$, $p(c|w)$ על פי הקורפוס
- נאותחל את $w|t$ על פי אחת הגישות הקודמות (ג, ד)
- o לולאת אימון [מספר פעמים, או עד שאיוני שימושוות]
 - נחשב [מחדרש] את $c|t$ על פי הנוסחה [ולא-or $w|t$] החדש
 - נחשב מחדש את $w|t$ על פי הנוסחה [ולא-or $c|t$] החדש

תוצאות ניסוחות: אתחול מודל מרקוב על פי גישה זו, נתן עבור אנגלית תוצאה שאינה פחותה ממודלים מורכבים הרבה יותר לא-מנוכה (בעברית זה לא כל כך שונה, כי ההסתמכוות על הטויות בגישה ד אפקטיבית ביותר).

הערה: בגישה זו, האלגוריתם עשוי לתת הסתברויות גדולות מ-0 לניתוחים/קטגוריות שלא הופיעו המילון עבור מילה נתונה.

לדוגמא, נניח כי במילון אין אפשרות של 'שם פרטי' [שם של בן אדם] עבור המילה 'אביב'. כלומר ההסתברות של אפשרות זו תאותחל ל-0:

$$p(\text{שם פרטי} | \text{אביב}) = 0$$

אם בתרבות הנוכחית, המאוחרת לכתיבת זמן המילון, כפי שהיא משתקפת בקורפוס העדכני, 'אביב' היה שם של אדם, ההקשרים בהם מופיע מילה זו יתאיםו לתבונת של שמות פרטיים. כך שההסתברות המחדשת שתוחשב לאפשרות הניתוח 'שם פרטי' ל'אביב' תגדל, ובפועל תתווסף עבורו אפשרות ניתוח חדשה.

מצד אחד זה טוב, מצד שני עשוי להיות פרוע (נוסף ניתוח חדש למילה, ובעקבות כך נוסף ניתוח חדש להקשרים שבהם היא מופיע, וחזר חלילה)

עיצוב ב Map-Reduce

האלגוריתם מבוסס על מעבר חוזר ונשנה של כל קומבינציות מילה/הקשר-קטgorיה, ולכן מומלץ לבצעו באופן מבוזר.

אתחול

- o חישוב $w|c|p$ – ע"פ הפרק בו חישבנו הסתברויות MLE
- o איתחול $w|t|k$ לפי אחת הגישות הקודמות (ג,ד)

לולאה

- o חישוב $c|t|k$ כמשימת Map-Reduce
- o חישוב $w|t|k$ כמשימת Map-Reduce

הקלט למשימות הלולאה

wc.txt

word context $p(w|c) p(c|w)$

tw.txt

word category $p(t|w)$

(חישוב $c|t|k$)

Class Mapper1

```
method Map(<c,w,t>, <ptw,pwc>)
    Emit(<t,c>, ptw · pwc)
```

Class Reducer1

```
method Reduce(<t,c>, values)
    sum := 0
    for value in values
        sum := sum + value
    Emit(<t,c>, sum)
```

tc.txt

context category $p(t|c)$

чисוב $w(t|p)$

Class Mapper2

```
method Map(<c,w,t>, <ptc,pcw>)
    Emit(<t,w>, ptc · pcw)
```

Class Reducer2

```
method Reduce(<t,w>, values)
    sum := 0
    for value in values
        sum := sum + value
    Emit(<t,w>, sum)
```

כיצד נdag לך ש Mapper2 יקבלו את הקלט המתאים למוגדרת ה map?
נשים לב, כי המידע שמגיע למוגדרת ה map מבוסס על נתונים בשני קבצים:

Mapper1: tw.txt, wc.txt

Mapper2: tc.txt, wc.txt

כלומר, נדרש שידוך של רשומות בקבצים שונים, ע"פ 'מפתח זר' (w,c,t)

[?]

- אפשרות א: נניח כי ניתן לאחסן בזיכרון את שני הקבצים הנדרשים בכל סיבוב (לא סקליבלי)
- אפשרות ב: נניח כי 'הקובץ הקטן' (wz בסיבוב הראשון, tc בסיבוב השני) נכנס לזכרון. כלומר
נדרש סיבוב מקדים לפני כל שימוש בלולאה של JoinSideJoin MapperSideJoin
- אפשרות ג: לבצע סיבוב מקדים לפני כל שימוש בלולאה של ReduceSideJoin

7. ניתוח תחבירי (Syntactic Parsing)

7.1 מבנה תחבירי

עד כה, זיהינו את הקטגוריה של כל מילה במשפט (פועל, שם, תואר...).

קיימים מקרים שבהם זיהוי הקטגוריות אינם מספק. לדוגמה:

דני אכל פיצה עם עגבניות

דני אכל פיצה עם חברים

למילים בשני המשפטים יש את אותן קטגוריות, אך המבנה של המשפטים (ומובנים) שונה:

דני [שם פרטי] אכל [פועל] פיצה [שם עצם] עם [מילת יחס] עגבניות [שם עצם]

דני [שם פרטי] אכל [פועל] פיצה [שם עצם] עם [מילת יחס] חברים [שם עצם]

במשפט הראשון, מילת היחס 'עם' מתייחסת ל'עגבניות'

במשפט השני, מילת היחס 'עם' מתייחסת לאכלי'

מלבד מציאת הקטגוריות של המילים, יש למצאו גם את היחסים בין המילים במשפט, כלומר מהו מבנה המשפט – **ניתוח תחבירי**.

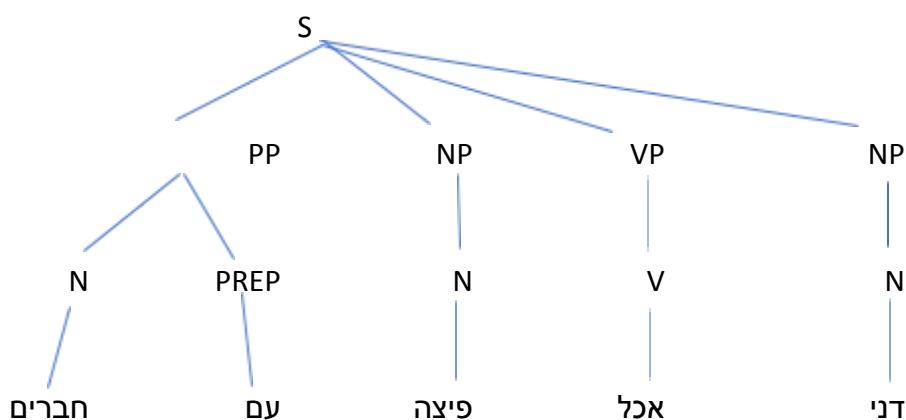
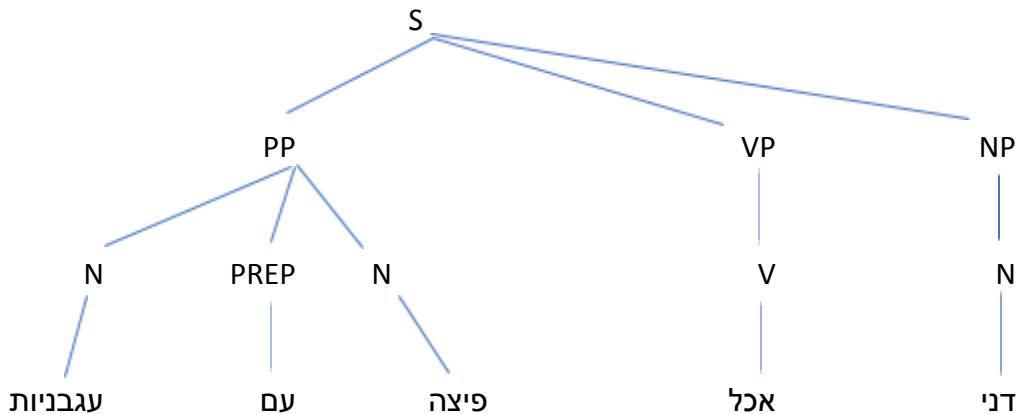
כיצד נציג/נمدל את המבנה של המשפטים בשפה?

קיימות שתי גישות:

1. מבנה פסוקיות (constituency structure)

המשפט מארגן במבנה של עצ, כאשר המילים הן העלים, והקודקודים השונים מייצגים את הקטגוריות של חלקו המשפט השונים.

לדוגמא: דני אכל פיצה עם עגבניות, דני אכל פיצה עם חברים



במודל זה, הגרף מתאר מצד אחד ('מלמטה למעלה') את היררכיה הקטגוריות של כל מילה, ומצד שני ('מלמעלה למטה') את תהליכי הגזירה של של המשפט, כולם את האופן שבו נוצר המשפט מהקטgorיה הכללית S.

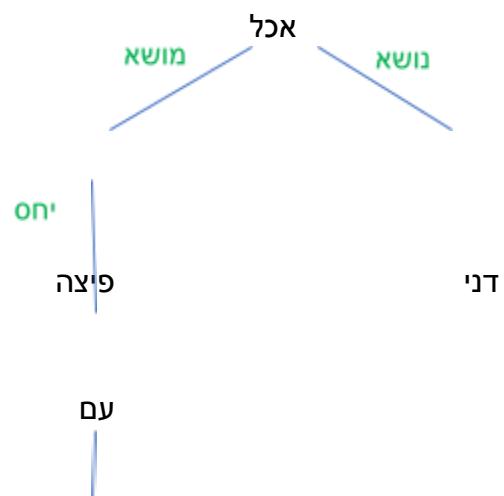
היצירה של המשפט במודל זה, הינה תוצר תהליכי מרקובי מבוסס על דקדוק חסר הקשר. דקדוק זה מגדיר את ההסתברויות לעבר מקטגוריה אחת לסדרת קטגוריות אחרות ('מטריצה A'), וכן את ההסתברויות לעבר מקטגוריה למילה ('מטריצה B').
לדוגמא:

```
S ⊥ NP VP PP [0.6]
S ⊥ NP VP NP PP [0.4]
NP ⊥ N [1]
VP ⊥ V [1]
PP ⊥ N PREP N [0.5]
PP ⊥ PREP N [0.5]

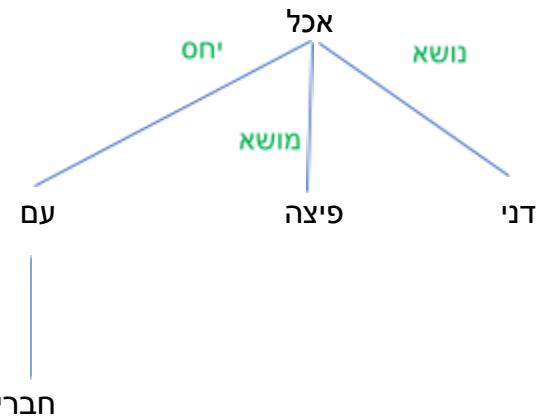
N [דני 0.4]
N [פיצה 0.4]
N [עגבניות 0.1]
N [חברים 0.1]
V [אכל 1]
PREP [עם 1]
```

2. מבנה תלויות (dependency structure)

המשפט מארגן במבנה של עצ: הקודקודים הם המילים, וצלע בין שתי מילים מצינית קשר תחבירי ביניהן.
לדוגמא:



עגבניות



במודל זה, המידע המרכזי המוצג הוא לא הקטגוריות של המילים (בדרכם כלל מצינים אותן בקודקוד לצד המילה), ולא האופן שבו נוצר המשפט, אלא היחסים בין המילים.

7.2 בעיית הניתוח התחבירי

נתון: משפט

יש למצוא את המבנה התחבירי שלו (ע"פ אחת הגישות)

7.2.1 אפליקציות

1. מציאת קבוצות מילים בעלי מונח משותף על פיו הסבירה התחבירית שלהם בקורסוס.

לדוגמא:

הלר
נסע
הגיע

לחיפה	דני
لتל אביב	יוסי
הביתה	האיש
	האישה

2. מענה לשאלות

מי היה בתל אביב השנה שעבירה?

דני נסע לתל אביב לפני חודש
יוסי י郎 לתל אביב אחר

Information Extraction .3

חילוץ תובנות של מידע, עובדות, ממגר עזום של טקסט.
כמו לדוגמה: חילוץ שלשות של נושא, נשוא, מושא.

CIA	train	exiles, agent, fighter, army
	use	building, missile...
	provide	lists, prof....

4. פישוט משפטי

ראיתי ילד שנouse לאט
ראיתי ילד. הילד nose לאט

5. תרגום

Fruit flies like a banana

6. גירעה טקסטואלית

נתונים שני משפטיים, יש לקבוע האם ניתן להסיק ממשפט אחד את המשפט השני.

לדוגמה: דני נסע ביום שלישי בשבוע שעבר לחיפה □ ישראלי הגיע בשבוע שעבר לצפון

7.2.2 שיטות

כיצד נקבע מהו המבנה התחבירי של משפט נתון?

1. מבנה פסוקיות

זכור, המשפט במודל זה הינו תוצר של תהליך מركובי המבוסס על דקડוק הסתברותי ('מטריצה A', 'מטריצה B').

?

- יש ללמידה את הדקડוק הסתברותי. הרחבה של אלגוריתם באום-וילש, EM.
- בהינתן דקડוק הסתברותי ומשפט, נ恢זר את התהילך שיצר משפט זה ('סדרת המצביעים', החוקים שנבחרו בתהילך) עם אלגוריתם ויטרבי.

ראינו כבר. לא מעניין. גם לא עובד הכל טוב. והמידע גם לא הכיכי אפקטיבי (לא ניתן לדוגמא לענות על שאלה פשוטה וסופר-חשובה: מי הנושא של המשפט)

2. מבנה תלויות

ניתן לחשב על הניתוח התחביבי במודל זה כבעית סיווג:

- נתון משפט, וכל העצים האפשריים עבורו (כל העצים האפשריים עברו גוף שהקווודים שלו הם המילימ'
- יש לבחור את העץ המתאים ביותר למשפט.

דיברנו כבר בתחום הקורס על בעיות סיווג. בבעיות הסיווג, נתון אובייקט ושאלתו לאן הוא הוא שייר. המسوוג הינו פונקציה המוצאת עבור כל אובייקט את מחלקת הסיווג שלו. המסווג נבנה בתהילך של אימון ע"פ דוגמאות מוכנות, בתהילך זה נמצא הקשר בין מאפייני האובייקטים למחלקת הסיווג שלהם. לדוגמא: סיווג פירות.

נרחיב מעט על בעיה זו.

7.2.2 בעית הסיווג

זכור, דיברנו על כך שבעית הסיווג מבוססת על שלושה שלבים:

1. ייצוג האובייקטים מהם אוסף המאפיינים המתארים כל אובייקט?

לדוגמה, עבור פרי: קוטר, משקל, מרקם, צבע

מאפיינים אלו מיוצגים על ידי וקטור (3, 0.3, 2.3, 17)

נסמן את הפונקציה ההופכת אובייקט לווקטור מאפיינים על ידי φ

הכנות אופי דוגמאות .2

יש להזכיר אוסף של זוגות של אובייקט (המיוצג על ידי וקטור מאפיינים) ומחלקה הסיווג שלו:
{<ψ(obj), class>}

3. אלגוריתם אימון

האלגוריתם מקבל את אוסף הדוגמאות, והוא לומד פונקציה המתקבלת וקטור מאפיינים של אובייקט ומיציריה את מחלקה הסיווג המסתברת ביותר עבורו.

יש בעולם אינסוף פונקציות...

נתמוץ ללימוד פונקציות לינאריות.

יתרה מכך, נצטמצם לפונקציה מהצורה: $f(x) = ax$

כasher x הוא וקטור מאפיינים, |x| הוא וקטור המציג את המשקל של כל מאפיין

$$f(x) = w \cdot x = w_1 x_1 + \dots + w_n x_n$$

7.2.3 הניתוח התחביבי (של מבנה תלויות) כבעית סיווג

צימנו כבר קודם: נתון משפט, וכל העצים האפשריים. יש לקבוע לאיזה עץ ('מחלקת סיווג') שיר המשפט. דומה לסיווג פירות, ארכוניה:

- כמוות מחלקות הסיווג במקורה שלנו עצומה (מספר העצים האפשרי מול מספר סוג הਪירות השונים) האובייקט עצמו - המיללים במשפט – דל במאפיינים פוטנציאליים (בניגוד למאפיינים רבים של הפרי).

האלגוריתם שנציג ([קולינס, 2004](#)) מתמודד עם שתי נקודות אלו:

- נראה להלן המאפיינים של משפט יכולו גם את מאפייני מחלוקת הסיווג שלו (כלומר מאפיini אחד העצים)

הרעין הכללי:

- אתחול
- o ווקטור משקלות 0
- o ייצור כל העצים האפשריים עבור כל משפט באוסף הדוגמאות, וייצוג כל עץ כווקטור מאפיינים.
- בכל איטרצית אימון
 - o עבור כל דוגמת אימון (משפט, כל העצים האפשריים עבורו, העץ שנבחר עבורו ידנית)
 - בחירת העץ המתאים למשפט ע"פ ווקטור המשקלות הנוכחי.
 - עוברים על כל העצים האפשריים עבור משפט זה
 - מחשבים את הפונקציה $\alpha = \phi(x)$ עבור ווקטור המאפיינים הנוכחי x
 - בחירת העץ שנותן את הערך הגבוה ביותר לפונקציה
 - השוואת הבחירה ע"פ המודל הנוכחי לעץ הנבחר ידנית
 - אם הוא שונה – תיקון ווקטור המשקלות בהתאם

```

Perceptron( { <Xt, Yt> } )
w0 = (0,...,0)
k = 0 // the current version of w
for n : 1 ... N // N training iterations
  for t : 1 ... T // T training examples
    y' := argmaxy wk φ(xt,y') // select the tree which maximizes the function according to W
    if (y' != yt) // the correct tree was not selected
      wk+1 = wk + φ(xt,yt) - φ(xt,y')
    k := k +1
  
```

בתום התהילה, יש ווקטור משקלות w .
 בהינתן משפט חדש לניתוח x , נחשב את הפונקציה $\phi(x, \alpha)$ עבור כל העצים האפשריים α למשפט, ונבחר את העץ שנותן ערך מקסימלי לפונקציה.

דוגמא:

נתונות שתי דוגמאות בקלט עבור תהליך האימון:

1X: ילדה אכלה תפוח
 1Y: ילדה → אכלה → תפוח

2X: סִפְרַ עָזָר
 2Y: סִפְרַ עָזָר

קביעת המאפיינים

נניח כי לכל מילה במשפט מצוינת בעץ גם הקטגוריה שלה

1X: יְלָדָה אֶאָכֵלָה תְּפֻוח
 1Y: יְלָדָה → אֶאָכֵלָה
 noun verb noun

2X: סִפְרַ עָזָר
 2Y: סִפְרַ עָזָר
 noun adj

נדיר את המאפיינים הבאים, על בסיס head, child, child {category X} (המנוחים head, child, child קשורים לתאוריה בלשנית על דבר 'יחס הכוח' בין המילים, איזו מילה 'שלוטה' ואיזו 'נשלטה', כפי שבא לידי ביטוי בכךון של הצלע המכונה בינהן)

head-noun
 head-verb
 head-adj
 child-noun
 child-verb
 child-adj

נייצר את כל העצים האפשריים לכל דוגמא, ובבנה כבר CUT את וקטור המאפיינים עבור כל אחד:

1X: הילדה אכלת תפוח
 11Y: יְלָדָה → אֶאָכֵלָה → תְּפֻוח (0,1,0,2,0,0)
 12Y: יְלָדָה → אֶאָכֵלָה → תְּפֻוח (1,1,0,1,0,0)
 13Y: יְלָדָה → אֶאָכֵלָה → תְּפֻוח (2,0,0,1,1,0)

2X: סִפְרַ עָזָר
 21Y: סִפְרַ עָזָר (1,0,0,0,0,1)
 22Y: סִפְרַ עָזָר (0,0,1,1,0,0)

натчил את иектора масколов:

$$w^0 = (0,0,0,0,0,0)$$

איטרציות האימון

$$N = 1$$

$$t=1$$

יש לבחור, על פי וקטור המשקלות הקיימים, את העץ שנותן הערך הגבוה ביותר עבור הפונקציה:

$$w^0 \varphi(x_1, y_{11}) = (0,0,0,0,0,0) \cdot (0,1,0,2,0,0) = 0$$

$$w^0 \varphi(x_1, y_{12}) = (0,0,0,0,0,0) \cdot (1,1,0,1,0,0) = 0$$

$$w^0 \varphi(x_1, y_{13}) = (0,0,0,0,0,0) \cdot (2,0,0,1,1,0) = 0$$

בפעם הראשונה כל העצים שוים כי וקטור המשקלות 0. נניח כי נבחר אקראיית העץ השלישי 13:

העץ הנבחר 13 שונה מהעץ שתויג ידנית עבור הדוגמא 13 \Rightarrow יש לתקן את וקטור המשקלות:

$$w^1 = w^0 + \varphi(x_1, y_1) - \varphi(x_1, y_{13}) = (0,0,0,0,0,0) + - (0,1,0,2,0,0) (2,0,0,1,1,0) = (-2,1,0,1,-1,0)$$

$$t=2$$

$$w^1 \varphi(x_2, y_{21}) = (-2,1,0,1,-1,0) \cdot (1,0,0,0,0,1) = -2 + 0 + 0 + 0 + 0 + 0 = -2$$

$$w^1 \varphi(x_2, y_{22}) = (-2,1,0,1,-1,0) \cdot (0,0,1,1,0,0) = 0 + 0 + 0 + 1 + 0 + 0 = 1$$

העץ הנבחר 22 שונה מהעץ שתויג ידנית עבור הדוגמא 22 \Rightarrow יש לתקן את וקטור המשקלות:

$$w^2 = w^1 + \varphi(x_2, y_2) - \varphi(x_2, y_{22}) = (-2,1,0,1,-1,0) + - (1,0,0,0,0,1) (0,0,1,1,0,0) = (-1,1,-1,0,-1,1)$$

$$N=2$$

...

בעיה חישובית: קיימים עצום של עצים אפשריים למשפט (אקספוננציאלי). לא ניתן באמצעות חישובית לבצע זאת שוכ ושוב.

פתרון: נבחר את העץ המתאים ביותר בגישה אחרת: העץ הנבחר הוא העץ ששילוב אינטואיטיבי שלו בלבד, נותן את הערך הגבוה ביותר. בגישה זו, המאפיינים הינם מאפיינים של צלע בודדת ולא של עץ שלם.

כלומר, נחליף את השורה

$$y' := \operatorname{argmax}_{y'} w^k \varphi(x_t, y')$$

בשורה:

$$y' := \operatorname{argmax}_{e \in V \times V} \sum_{e \in E} w^k \varphi(e)$$

בנוסחה זו, עוברים על כל קבוצות הצלעות האפשרות ומגדירים את האיכות של הקבוצה ע"פ סכום ערכי הפונקציה על כל מאפייני צלע בודדת בנפרד.

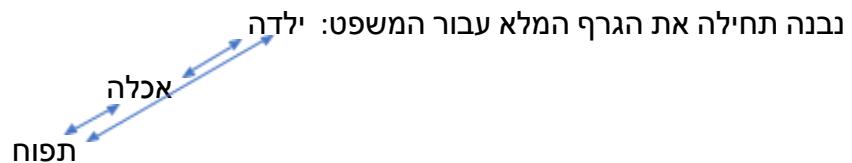
מדובר למעשה בבחירה עצם פורש מקסימלי לגראף ממושקל (נדגים מיד) – ניתן לבצע אותה ב $O(n^3)$, עבור ח' קודקודים.

דוגמא:

המשפט: הילדה אכלה תפוח

ווקטור המשקלות: $(1,0,0,2,3,0)$

יש למצוא את העץ המסתבר ביותר על פי ווקטור המשקלות הנוכחי.



نبנה תחילה את הגראף המלא עבור המשפט: ילדה
תפוח
אכלה

نبנה לכל צלע בנפרד ווקטור מאפיינים, ונמשקל את הצלע ע"פ ערך הפונקציה עם ווקטור המשקלות הנוכחי:

ילדה \oplus אכלה $(1,0,0,0,1,0)$

$$w\varphi(e_1) = (1,0,0,2,3,0) \cdot (1,0,0,0,1,0) = 4$$

ילדה \oplus תפוח $(1,0,0,1,0,0)$

$$w\varphi(e_2) = (1,0,0,2,3,0) \cdot (1,0,0,1,0,0) = 1$$

אכלה \oplus ילדה $(0,1,0,1,0,0)$

$$w\varphi(e_3) = (1,0,0,2,3,0) \cdot (0,1,0,1,0,0) = 2$$

אכלה \oplus תפוח $(0,1,0,1,0,0)$

$$w\varphi(e_4) = (1,0,0,2,3,0) \cdot (0,1,0,1,0,0) = 2$$

תפוח \oplus ילדה $(1,0,0,1,0,0)$

$$w\phi(e_5) = (1,0,0,2,3,0) \cdot (1,0,0,1,0,0) = 3$$

תפוח ↗ אכליה (1,0,0,0,1,0)

$$w\phi(e_6) = (1,0,0,2,3,0) \cdot (1,0,0,0,1,0) = 4$$

נמצא את העץ הפורש המקיים מלאי עבור הגרף הממושך המלא שיצרנו.

חיסרונות: וקטור המאפיינים מבוסס כל פעם רק על צלע אחת, ולא על כל העץ. מצמצם מאוד את אפיון המשפט. לא ניתן לדוגמא להגדיר מאפיין של 'לכמה קודקודים יש בן מסווג שם עצם ובן מסווג תואר' ('יהיה קשה יותר להבחין בין 'אכל פיצה עם חברים' ל'אכל פיצה עם עגבניות')

☒ נחזור להערכת הקודמת על בסיס כל העצים האפשריים, ובוצעו את האלגוריתם בתבנית `map-reduce`, כך שרמת המקובל יכולה להגיע למצב אחד מעבד דוגמא אחת בלבד באיטרציה אחת. נשים לב לכך, כי המקובל נדרש כאן לא בשל הכמות העצומה של המידע בקלט (אין הרבה דוגמאות), אלא בשל סיבוכיות האלגוריתם.

יעצוב Map-Reduce עבור אלגוריתם Perceptron ללמידה ווקטור משקלות עבור הניתוח התchapiri'

הרענון הכללי:

- כל איטרציה אימונן היא סיבוב אחד של R-M
- כל Mapper מקבל רק חלק מהדוגמאות (ברמת המקובל הגבוהה ביותר, על Mapper מקבל דוגמא אחת בלבד, כחומר ספליט הכלול דוגמא אחת).
- בשלב ההתחול, הMapper קורא את וקטורי המשקלות האחרונים (מהאיטרציה הקודמת)
- מתודת `map` מקבלת דוגמא (משפט, והעץ הנכון עבורו), מרים את העיבוד של הדוגמא ע"פ האלגוריתם (מציאת העץ המתאים מבין האפשריים והשוואתו לעץ הנכון), וمعدכנת מקומית את וקטורי המשקלות.
- בסוף משימת Mapper, ווקטור המשקלות המקורי, שעודכן ע"פ הדוגמאות שהגיעו למשימה זו, ישלח לרדיוסר.
- הרדיוסר (היחיד) ימזג את כל וקטורי המשקלות שהוא קיבל, מקבוצות הדוגמאות השונות. וכותב את וקטורי המשקלות לשיבוב הבא.

Class Mapper

Method Initialize

$w^0 := \text{loadLastW}()$

$k := 0$

```

Method Map( $x_t, y_t$ )
     $y' := \operatorname{argmax}_{y'} w^k \phi(x_t, y')$ 
    if ( $y' \neq y_t$ )
         $w^{k+1} = w^k + \phi(x_t, y_t) - \phi(x_t, y')$ 
     $k := k + 1$ 

```

```

Method Close()
    Emit("w",  $w^k$ )

```

Class Reducer

```

Method Reduce(key, Ws)
    Wsum := (0, ..., 0)
    for W in Ws
        Wsum := Wsum + W
    Emit("W", Wsum)

```

מה אנחנו מפסידים בעיצוב המקביל, זהה? באלגוריתם המקביל, נקודת הבסיס לעיבוד דוגמא מסוימת היא וקטור המשקלות מהאיטרציה הקודמת. בעוד שבאלגוריתם הסדרתי, וקטור המשקלות הבסיסי לכל דוגמא היה הווקטור שנלמד על ידי הדוגמא הקודמת באיטרציה הנוכחית. האם זה משנה? עד כמה?

2010. Mc Donald et al. בדקו שאלה זו (במסגרת הצעת עיצוב R-M שראינו זה עתה) ומצאו כי ה'ഫוד' אינו מרובה:

אחוזי הדיוק של מסוג שאומן באלגוריתם הסדרתי: 84.7%
 אחוזי הדיוק של מסוג שאומן באלגוריתם המקביל: 84.5%
 בניסוי שלהם, רמת המקובל לא הייתה דוגמא אחת לMapper אחד, אלא כל מאפר קיבל קבוצת דוגמאות. מענין יהיה לבדוק, מה יהיו התוצאות ברמת המקובל המksamלאית (משפט פר מאפר).

הערה: מודל גנרטיבי מול מודל דיסקרימינטיבי

בשתי הבעיות האחרונות פגשנו שני סוגי של מודלים.

בעיית התיאוג (מציאות הקטגוריות של המילים) – מודל מרקוב: מודל גנרטיבי

- ראיינו את הטקסט כתוצר של תהליך, פלט של מכונה
 - אימון: מציאת הפרמטרים הסמויים של המודל/המכונה (המטריצות B,A)
- הפונקציה הנלמדת היא פונקציית הסתברות ייצור הפלט, כולם פונקציית ה joint:

$$P(X,Y) = P(Y|X)P(X)$$

במודל מركוב נדרשנו לדוגמא למדוד את הסתברות המצב ואת ההסתברות לייצר ממנו פלט.

- ניתוח: שחזור התהיליך שייצר את הטקסט, מה קרה במכונה (מהי סדר המצביעים/הקטגוריות
משמעותה הכי טוב את המשפט שנוצר)

בעית הניתוח התחבירי – מסעוג: מודל דיסקרימינטיבי

- הטקסט הוא לא פלט של מכונה, אלא קלט שצריך למצוא את תכונתו (במקרה שלנו, צלעות בין
מילים).
- אימון: מציאת פונקציה הקושרת בין הקלט/הטקסט לתכונותיו/מחלקות-הסיווג (העז המתאים)
הfonקציה הנלמדת היא פונקציית ההסתברות המותנית $A|Y|P$). בהינתן אלמנט X מה ההסתברות
שמאפיינו ה Z . בדרך כלל בוחרים פונקציה כלשהי 'מקובלת' ולומדים מהדата את הפרמטרים
שליה. כלומר מוקסמים באימון את $X|Y|P$).
- באימון המנתח התחבירי לדוגמא בחרנו בפונקציה הלינארית $Aw=f$ ולמדנו את פרמטר
המשקלות w .
- ניתוח: הפעלת הפונקציה שנלמדה על הקלט (הטקסט) לשם קבלת תכונתו (העז)

8. סיווג מסמכים

בפרק זה, נבחן מספר שיטות לחלוקת קבוצות מסמכים לקבוצות, כאשר לכל קבוצה מסמכים יש מאפיין
דומה. אף ננסה לאפיין את הנושא' שבו עוסקת כל קבוצה מסמכים.

שימושים:

- מנוע חיפוש המאפשר חיפוש ע"פ נושאים. במנוע שכזה, התוצאות יהיו מסמכים העוסקים
בנושאים העולים מהשאלתא.
- ייצרת מאגר הומוגני ללמידה בהמשך. קיימים אלגוריתמי למידה העובדים טוב יותר כאשר הקלט
מורכב מסמכים בעלי מאפיינים דומים (אולי נראה כזה בהמשך).
-

שיטות:

- האם מסמן שיר לנושא אחד בלבד (hard), או שהוא תמהיל של נושאים (soft)
- האם יש מידול של הנושאים (topic modeling) או רק חלוקה לקבוצות (clustering)
- האם הלימוד הוא מונחה (supervised), לימוד מדוגמאות מנוטחות (דנית) או שהוא אינו מונחה
(unsupervised), לימוד מהטקסט הנוכחי לכשעצמו)
- מודל גנרטיבי או מודל דיסקרימינטיבי

אנו נתמקד בכמה שיטות קונקרטיות:

- K-means: hard, clustering, unsupervised
- LSI: soft, unsupervised, clustering

- LDA: soft, topic modeling, unsupervised
- רשת נירונית: soft, topic modeling, supervised

K-Means 8.1

אלגוריתם K-means מחלק קבוצת מסמכים נתונה ל K קבוצות (כאשר ה K הוא פרמטר הנבחר ע"י המשתמש), כאשר כל מסמר שייך לקבוצה אחת.

הרענון הכללי:

- o מסמר מיוצג ע"י וקטור מאפיינים
- המאפיינים פשוטים ביותר יהיו מספר מופע המילים השונות בשפה במסמן הנתון.
- ניתן לכלול במאפיינים גם את הקטגוריות של המילים (פועל, שם עצם)
- ניתן לכלול במאפיינים גם קשרים תחביריים (מושא, מושא,...)
- ניתן לכלול גם מאפיינים שהם שילובים של מאפיינים: זוגות מילים, מילה-קטgorיה...
- o קבוצת מסמכים מיוצגת ע"י וקטור (בגודל של וקטור המאפיינים), אשר הווקטורים של המסמכים השיכים לקבוצה זו קרובים אליו אלגברית (הוא נמצא במרכז, ועל כן הוא מכונה *centroid*)
- o האלגוריתם
 - נבחר אקראית K וקטורי מרכז (מרכזואידים), כאשר כל אחד מהווקטורים מייצג קבוצת מסמכים.
 - משיכים לכל צентрואיד את המסמכים הקרובים אליו, ע"פ המרחק ביניהם (למרחוק וקטורי)
 - נגזר מחדש את הקורדינטות של כל צентрואיד, כך שהוא עומד במרכז של קבוצת המסמכים ששוכנה לו.
 - ו חוזר חלילה, עד שאין שינוי

אלגוריתם Lloyd

- o נתונים חובייקטיבים לקלסטור, המיוצגים ע"י וקטורי מאפיינים: $x_1 \dots x_n$
- o אתחול: בחירה אקראית של K צентрואידים $m_1^0 \dots m_K^0$
- o לולאה
 - שיר כל מסמר לצентрואיד הקרוב אליו ביותר אלגברית

$$S_i^t = \{x_p : \operatorname{argmin}_i \operatorname{distance}(x_p, m_i^t)\}$$
 - חישוב מחדש של קורדינטות הצентрואידים, כך שיימדו במרכז המסמכים שלהם

$$m_i^{t+1} = \frac{\sum_{x_j \in S_i^t} x_j}{|S_i^t|}$$

עיצוב ב R-M

- o כל איטרציה אימונן היא סיבוב של R-M
- o המאפר מבצע את השלב הראשון – שיר מסמן לצנטרואיד
 - מקבל מסמן ובודק למי הוא שיר, ושולח את הזוג קבוצה, מסמן לדיזור
- o הדיזור יקבל עבור קבוצה מסוימת את כל המטכרים השייכים לה, ויכיל מחדש את קורדיננטת הקבוצה בהתאם.

Class Mapper

Method Initialize

```
centroids := getLastCentroids()
```

Method Map(docId, docVect)

```
centroidId := -1
min := infinity
for centroid in centroids
    dist := distance(docVect, centroid)
    if (dist < min)
        min := dist
        centroidId := centroid.id
Emit(centroidId, docVect)
```

Class Reducer

```
method Reduce(centId, docVets)
    size := 0
    newCentroid := (0,...,0)
    for docVect in docVets
        newCentroid := newCentroid + docVect
        size := size +1
    for i :=1 to newCentroid.length
        newCentroidi := newCentroidi / size
    Emit(centId, newCentroid)
```

(Latent Semantic Indexing (LSI 8.2

בגישה זו נבצע soft-clustering של מאגר טקסטים ל K קבוצות/נושאים (ambil לאפין את הנושא של כל קבוצה). כמו K-Means רק שמסמן שיר, בהסתברות שונה, לכל הנושאים.

נקודות המוצא: מטריצה C הקובעת את מידת השיכנות של כל מילה במאגר לכל אחד מהמשמעותים (קיימות גישות שונות לקביעת מידת שיכנות זו, נראה בפרק הבא את המדריך הקליאטי tf-idf הקובע את מידת ההתאמנה של מילה למסמך, על פי השכיחות היחסית של מילה זו במסמך)

C

	a.txt	b.txt	c.txt
קנייה	0.2	0.4	0.1
מכירה	0.5	0.4	0.1
כיסא	0.1	0	0.7
שולחן	0.2	0.2	0.1

ברצוננו להגדיר מטריצה קטנה יותר, עם פחות שורות (למעשה עם K שורות), בה המידע במטריצה המקורית 'מהודק' יותר. כלומר, מידת ההתאמנה של מילים שונות למשמעות, יהפכו למידת ההתאמנה של 'מושגים' למשמעותם בלבד.

	a.txt	b.txt	c.txt
[מושחר]	0.7	0.8	0.2
[ריהוט]	0.3	0.2	0.8

בגישה זו פועלה זו נעשית באמצעות אלגבריים טהורים:

פורמלית

נתון מאגר של p מושגים, ו v מילים שונות (בדוגמא שלנו $v=4$, $p=3$)

נגדיר את המטריצה C , בגודל $p \times v$, כמייצגת את מידת ההתאמנה בין כל מילה v לכל מסמך C_i :

(tf-idf או כל מדריך אחר)

קיימת פועלה אלגברית המכונה SVD (Singular Value Decomposition), המפרקת מטריצה נתונה לשולש מטריצות, באופן הבא:

$$C = U_0 \Sigma_0 V_0$$

כאשר:

U_0 היא מטריצה בגודל $v \times v$

Σ מטריצה מלכנית אלכסונית בגודל $p \times v$ (מחוץ לאלכסון הכל 0, שורות/עמודות)

V_0 היא מטריצה בגודל $p \times p$

Σ כוללת את כל הערכים העצמיים של C בסדר יורדי ($\sigma_{i,j} > \sigma_{i,j+1}$ לכל i, j)

U_0 אורטורוגונאליות:

$$V_0^\top V_0 = I$$

$$U_0^\top U_0 = I$$

[מה שפורה באחת זו עמודה בשניה]

מתודתLSI בונה את המטריצה המבוקשת C' ממטריצות אלו באופן הבא:

- לוקחים את k הערכים העצמיים הగבוהים ביותר מ Σ ומضافים את השאר.
- o כך יהיו k-v שורות עם אפסים (בשל האלכסוניות של Σ המקורי) ו-k-d עמודות עם אפסים (עקב בחירת k ערכים עצמים בלבד)
- זה יגרום לאיפוס השורות והעמודות המקבילות ב V_0 (מה שהיא מתאפס אם היוו מכפילים אותן ב Σ החדשה):
- o במטריצה U יתאפסו k-v עמודות
- o במטריצה V יתאפסו k-d שורות
- לאחר מחיקת שורות ועמודות שכולן אפסים מקבילים: מטריצה U בגודל Axk, מטריצה Σ בגודל kAx, מטריצה V בגודל Pxk
- נגידר את C' להיות: $V \Sigma U = C'$.

C' היא בגודל Pxk כמו C המקורי, אך דרגתה היא k. לומר יש בה רק k וקטורים שאינם תלויים ליניארית. על פי משפט אקרט-יונג, C' היא הקרוב הטוב ביותר ביותר של מטריצות מדרגה k למטריצה C.

ניסויית, בהתאם לשיפור אונושי, התגלה כי C' מייצגת טוב יותר דמיון בין מסמכים (המරחק בין הייצוג הווקטור שליהם כטור בטבלה) ובין מילים (המරחק הייצוג הווקטור שלהן כפורה בטבלה)

ניתן להגדיר באופן מפורש מטריצות עם k שורות/עמודות באופן הבא:

$$\Sigma = D, \text{ בגודל } Pxk$$

$$U = W, \text{ בגודל } Axk$$

- המטריצה D נותנת למעשה soft-clustering של כל מסמר ל-k 'נושאים', המוטיבציה שלנו בפרק זה.
- מלבד זאת, ניתן להשתמש ב-D כדי לעירר את מידת הדמיון בין שני מסמכים על פי וקטורם 'הנושאים' שלהם.

D

	a.txt	b.txt	c.txt
['מסחר']	0.7	0.8	0.2
['ריהוט']	0.3	0.2	0.8

- המטריצה W מתארת למעשה soft-clustering של כל מילה ל-k 'נושאים'.
- מלבד זאת, ניתן להשתמש ב-W כדי לעירר את מידת הדמיון בין שתי מילים על פי וקטורם 'הנושאים' שלהם.

בניגוד למודל הסטנדרטי של *distributional similarity*, בו מוצגת כל מילה על ידי וקטור דليل עם מאפיינים רבים (המילה לפניה, המילה לאחריה, קומבינציות שונות) akan הווקטור קטן בסדרי גודל – המאפיינים הם K ה'נושאים'. יואב גולדברג ורועי לוי הראו כי גישה זו משיגה את אותן תוצאות של word embedding (להלן).

W

	[מ媾]	[ריהוט]
קניה	0.8	0.2
מכירה	0.9	0.1
כיסא	0.3	0.7
שולחן	0.4	0.6

חיפוש מילה מבוסס 'נושאים'
בהינתן מילה לחיפוש w , השורה W מגדירה את התפלגות הנושאים למילה. מידת ההתאמה של כל נושא לכל מסמרק מוגדרת ע"י המטריצה D. לעומת מידת ההתאמה של המילה w למסמרק j היא:

$$\sum_k W_{i,k} D_{k,j}$$

בעיה: פועלות ה SVD כבده מאד, בפרט עבור מטריצות גדולות (ביג-דאטא, הרבה מסמכים, הרבה מילים)
 █ נמקבל אותה
בעיה: לא ידוע על אלגוריתם מקבילי לבעה זו.

בל' קשר לשאלת המקובל, (Gao & Zhung (2005) Ci שיטת LSD אפקטיבית הרבה יותר אם הקורפוס המקורי הומוגני יותר (המסמכים סובבים סביר מספר קטן של נושאים).
 במידה והקורפוס המקורי מגוון מדי, הם מציעים תחילה לבצע קלואסטרינג קשיח שלו לקבוצות (נניח עם k-means) ולאחר מכן להפעיל את LSD בנפרד על כל קבוצה.
 באופן זה, מסמרק הקשור תחילה לקבוצה אחת, וב奏ור קבוצה זו להתפלגות של 'נושאים'.
 ניתן להמשיך כיוון זה, ולפתור בעזרתו גם את הבעיה החישובית של חוסר המקובל:

– נחלק את הקורפוס ל K קבוצות עם LSD-K-means על ידי Map-Reduce
 – נפעיל את LSD סדרתי על כל קבוצה, במקביל (לא קריטי, אין big data של קבוצות). באופן זה
 כאשר אוסף המסמכים קטן יותר וה SVD פחות כבד.

(Latent Dirichlet Allocation (LDA 8.3

למעשה מבוצע soft clustering (כל מסמרק ישoir בהתפלגות לכמה קבוצות), תוך אפיון הנושא של כל קבוצה.
 המודל הפעם גנרטיבי (ומכאן גם unsupervised, נלמד רק מהטקסטים). ע"פ מודל זה, המסמכים במאגר נוצרו ע"י התהיליך הבא:

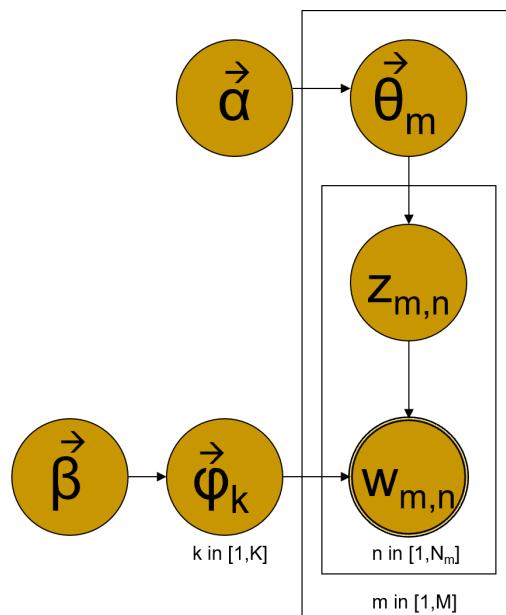
בהתנתק שיש K נושאים:

- עברו כל נושא
 - o בחר את תמהיל המילים בנושא זה, כולם התפלגות הקובעת מה הסתברות של כל מילה להיות שייכת לנושא (מה הסתברות ש'cosa' שייך ל'ספורט', מה הסתברות ש'שולחן' שייך ל'ספורט'). ע"פ התפלגות דריכלה (המייצרת התפלגות של התפלגות (מולטינומיות), ובמקרה שלנו התפלגות מילים לכל אחד מ K הנושאים)
- עברו כל מסמך לייצור
 - o בחירת תמהיל הנושאים במסמך, כולם התפלגות הקובעת מה הסתברות של כל נושא במסמך זה (עד כמה המסמך עוסק ב'ספורט', עד כמה הוא עוסק ב'כלכלה'), ע"פ התפלגות דריכלה (המייצרת התפלגות (מולטינומית), ובמקרה שלנו התפלגות נושאים לכל אחד מ M המסמכים)
 - o בחירת כמות המילים במסמך (ע"פ התפלגות פואסון)
 - o עברו כל מילה במסמך
 - בחירת נושא למילה ע"פ תמהיל הנושאים של המסמך
 - בחירת מילה עברו הנושא הנבחר מתוך תמהיל המילים לנושא זה

ניתן לתאר זאת סכמטית, כך (α , β הם פרמטרים של הדראיכלה):

K topics
M documents
 N_m words in document N

```
for all topic k in [1,K] do
    sample mixture components  $\phi_k \sim \text{Dir}(\beta)$ 
for all documents m in [1,M] do
    sample mixture proportion  $\theta_m \sim \text{Dir}(\alpha)$ 
    sample document length  $N_m \sim \text{Pois}(\xi)$ 
    for all words n in [1, $N_m$ ] in document m do
        sample topic index  $z_{m,n} \sim \text{Mult}(\theta_m)$ 
        sample term for word  $w_{m,n} \sim \text{Mult}(\phi_{z_{m,n}})$ 
```



אימון המודל

בשורה התחתוננה: יש ללמד מה הנושא של כל מילה בכל מסמך - המטריצה $\{Z\}_{d,w} = \{z_{d,w}\}$

Gibbs Sampling היא טכניקה לאימון שכזה.

מבנה נתונים: נתחזק את טבלאות המונים הבאות:

$N_{w,k}$ מספר הפעמים בהם נגזרה המילה w מהנושא k

$N_{d,k}$ מספר הפעמים בהם נושא k הופיע (כמילה) במסמך d

N_k מספר המילים הכלול בנושא k (כלומר מספר הפעמים בהם הופיע הנושא k במסמכים השונים)

אתחול:

ו בבחירה רנדומית של מטריצה Z (הצבות הנושאים השונים לכל מילה בכל מסマー), ואתחול המונים בהתאם.

- למעשה מדובר בבחירה נושא לכל מילה במסמך U^d ה Z הראשון (הrndom)

lolat האימון:

ו לכל איטרציה אימון

▪ לכל מסマー d

• לכל מילה במסמך w

ו הקטינה ב-1 של המונים עבור המילה והנושא שלה U^d Z הנוכחי

($Z_{d,w}$). ככלומר, אם נושא המילה הוא k (כלומר $k = Z_{d,w}$), אז נקטין

$N_{d,k}, N_{w,k}$, N_k

ו בחירת נושא חדש k' למילה w במסמך p , U^d התפלגות הנושאים למילה הנוכחית במסמך, כאשר ההסתברות לכל נושא ניתנת U^d נסוכה הבהא, בהתאם לטבלאות המונים המעודכנות (W הוא מספר המילים השונות):

$$[**] P(k) = \left(n_{d,k} + \alpha_k \right) \frac{n_{k,w} + \beta_w}{n_k + \beta \cdot W}$$

ו השמת הנושא החדש k' של המילה במטריצה Z : $Z_{d,w} := k'$

ו העלאת המונים של k' ב-1: $N_{d,k'}, N_{w,k'}, N_k$.

עיצוב מקבילי של אלגוריתם האימון בתבנית R-M

מתודת map: מקבלת מסマー ומפעילה עליו את גוף האלגוריתם. בסוף המשימה נשלחים המונים.

מתודת reduce: מקבלת את ערכי המונים בעקבות העבודה של כל אחד מהמסמכים ומחברת אותם.

כמו באימון המסוג, החיסרון הוא שנקודות המוצא של כל מסマー מתייחסות לעיבוד של המונים באותו הMapper על בסיס האיטרציה הקודמת.

הנחות על הזכרון:

- לא ניתן לאחסן בזיכרון את מטריצה Z כולה (גודלה הוא כגודל הקורפו)
- מצד שני, אם כל Mapper עובד על אוסף קבצים אחר, הוא צריך רק את העמודות במטריצה הנוגעות לקבצים שלו.

- ניתן להימנע מלהשתמש במטריצה Z כמבנה נתונים ולהסתפק רק במונחים: כאשר נדרש לדעת מה הנושא של מילה במסמך, נגזרל על פי ערכי המונחים את הנושא (כפי שזה נעשה כאשר משתמשים את הנושא למטריצה). **אלגוריתם זה שונה מהאלגוריתם המקורי:** הנושא של כל מילה בכל קובץ נקבע באופן דינמי בהתאם לערכי המונחים כתעט, ולא על פי ערכי המונחים בסוף איטרציית האימון לאחרונה כמו באלגוריתם המקורי.

- באופן זה כל Mapper ישמור בזיכרון

- $K: NK$ מספרים o
- $W: NWK$ מספרים (ו הוא מספר המילים השונות, לא מספר מופיעי המילים הכללי)
- $D_m: NDK$ מספרים, כאשר D_m הוא מספר המסמכים בסופלית של המאפר הנוכחי

Class Mapper

```

method Initialize
    load last  $ND_mK$  (only for the files of this mapper split) ,NWK,NK

method Map(docId, doc)
    for word in doc
        topic := pick topic according to counters  [**]
         $NDK_{doc,topic}--$ ;  $NWK_{word,topic}--$ ;  $NK_{topic}--$ ;
        topic' := re-pick topic according to updated counters  [**]
         $NDK_{doc,topic'}++$ ;  $NWK_{word,topic'}++$ ;  $NK_{topic'}++$ ;

method Close()
    Emit("k", NK)
    for each word w
        Emit("w-" $+w$ ,  $NWK_w$ )
    for each document d
        Emit("d-" $+d$ ,  $NDK_d$ )|

```

Class Reducer

```

method reduce(key, vectors)
    merged = new Vector()
    for v in vectors
        add(merged,v)
    Emit(key,merged)

```

ಅಪ್ಯಾನ ನೋಡು ಉದ್ದೇಶದಲ್ಲಿರುತ್ತವೆ

הנושא מאופיין סתם ע"י אינדקס (מספר מ 1 עד K). ניתן לאפין את הנושא על ידי המילים השכיחות בו ביותר (נניח 5 המילים השכיחות בו ביותר). מסתבר שהה די אפקטיבי.

לדוגמא, הרצה של אלגוריתם על משנה תורה לרמב"ם, נתנה בין היתר את 'ನೋಡು' הבאים:

- 9** נר הקליק פטילה שם מדליק המשמש איש לילה קל בדק
- 1** שלם בעל האיק מיב שור נוק פטר בהמה נוק בור
- 2** עשה חל חלה הפריש כמה שיעור עשה מין האטרוף פת
- 3** געשה פסל עין אכל מעט לש תורה הפסיד חשב קרבן
- 4** חכם פלאייד רב ישב בבוד עמד חכמה עם נכנס התיר
- 5** תרומה פרם חול הפריש מעשר זר כהן בגן טמא פרי
- 6** גזש קוש קדושים מקוש איש ספק פרט אמר גמור התקdash
- 7** ראייה בכיא חזקה שנח חזקן טון בעל אב נכס
- 8** האטרוף יד אכל שמר נשאר שאר עצם גראר מכאן חבר
- 9** היפר אב שמיע בעל נדר רשות יומ קנים אروس העלה
- 10** כסף מעה פקה סלע חול פרי פרט הוסף חיל עלה
- 11** שמר שואל מות שאל הפקיד שמייה חזיר שלח אנס נגנוב
- 12** מדרס לש עוזר טפח מטה ראיוי משכוב נגנ אנה טקאה
- 13** לקה הכה רדה נן ברת עשה אסור עבר מיטה מלכות
- 14** מת קרע אבל מטה קרוב שבע אב אפה אחר קבר
- 15** משללין פטר שבת משל אחר מפרק מהווים גבורה נודע מים
- 16** אב בן מת יורש אח בת אפה נכס נטול מלך

אותו מבנה נתונים, מאפשר גם למשמן חיפור מונחה-נושאים. במנוע חיפור שכזה, לא מחפשים את המסתמכים הכללים את המילים בשאלתא, אלא את המסתמכים העוסקים במילים בשאלתא (גם אם אין מופיעות בו), או חלוקת המסתמכים הכללים את מילות החיפור על פי הנושאים שלהם. לדוגמה, חיפור המילה 'שור' ברמב"ם, מניב את התוצאות הבאות:

1	שלם בעל חזק שור נזק פטר בהמה נזק בודר	0.0210184182015
32	הכיה עלה פריש שכח שור גדול קבע הניה הבעה יפה	0.0136645962733
65	בשר בהמה כי עוף דם חלב המר אסור אבר טמא	0.00293809024134
105	העללה ראה מלך ידע ברת ראייה לש שיש מקום זרך	0.00232867598137
74	קדש הקדיש מעיל מטבח בית הקדרש בזק פזה גננה בהמה	0.00231749710313
46	פרק הילכה טען ספר שיש בהמה חמשה נגמר סליק דקה	0.00210970464135
47	מים בכור זכר ילד בהמה נקבה פזה בעל פה נולד	0.00189753320683
108	מיאא החזיר בעל סיון אבד הכריז נפל מזא נטול אבבה	0.0017667844523
110	אליה קרבון הכיה פסאה שלם דם אמר הפריש נפל בהמה	0.00163755458515
117	שפט שחיטה שומט סכין שני שער פסה בגליה ידע חבורה	0.00105988341282

8.4 רשות נירוניים

הלימוד הפעם מונחה (כלומר, הקלט הוא זוגות של <מסמרק, נושא/ים>)

8.4.1 מבוא

אם הקלט מתויג, כלומר אנו יודעים מה הנושא/ים של כל מסמרק, ניתן ללמידה מסווג (כמו בפרק הקודם).

- ניצג כל מסמרק כוקטור מאפיינים (נכיה מה המילים המופיעות בו, או גם הקטגוריות שלהן, וכו')
- נרץ אלגוריתם אימון מסווג כלשהו, שילדmad פונקציית סיווג.
- בהינתן מסמרק חדש, ניצג אותו כוקטור על פי מאפייניו. ונפעיל את פונקציית המשווג כדי לקבל התפלגות נושאים עבור מסמרק זה.

חסרוןנות:

- לא ניתן להרכיב מאפיינים

לדוגמא:

- o אם המאפיינים הם המילים, המודל לא יודע להרכיב זוגות ושלשות של מאפיינים/מיללים. כדי לעשות זאת, אנו צריכים להגדיר עצמנו מאפיינים נוספים עם הקומבינציות (כל המילים הבודדות, כל זוגות המילים, כל שלשות המילים – ווקטור מאפיינים עצום)
- o אם המאפיינים הם המילים וקטגוריות, המודל לא יודע להרכיב מילים וקטגוריות. כדי לעשות זאת, אנו צריכים להגדיר עצמנו מאפיינים נוספים עם הקומבינציות (כל המילים הבודדות, כל הקטגוריות הבודדות, כל שילוב של מילה-קטgorיה)
- יש רק פונקציית סיווג אחת, לא ניתן לשלב ייחודי מספר פונקציות.
- יש לשפר את איכות המשווג

■ רשותות נירוניות

2.4.2 תאור המודל

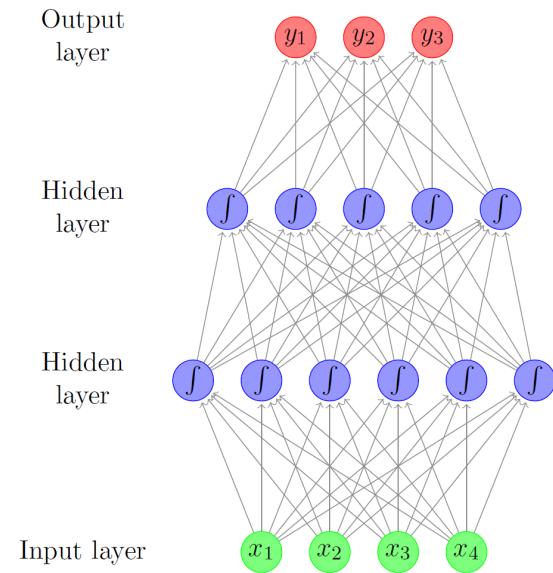
כללי: רשות ניירונית הינה מודל המתאר את הקשר בין אובייקט הקלט, המיצג סדרה של ווקטורים, לבין הפלט המתאים לו, המיצג אף הוא סדרה של ווקטורים, על ידי רשות של פונקציות המקבלות כפרמטר מטריצות.

לומר, המודל עשיר מבחינות ייצוג הקלט והפלט, וכן מבחינות הפונקציה המקשרת ביניהם. רשות הפונקציות והפרמטרים ביניהם, מאפשרים לתאר טוב יותר הן את הייצוג של הקלט והפלט והן את הקשר ביניהם.

ייצוג: כל מאפיין הינו ווקטור (בניגוד למסוג הרגיל, בו כל מאפיין היה ערך אחד בווקטור המאפיינים). גודל הווקטור נקבע על ידי מוצב הרשות.
לדוגמא: כל מילה במסמך תייצג ע"י ווקטור שלם. מסמך ייצג על ידי הווקטורי של כל המילים המופיעות בו. בפרט, ניתן לקבוע שווקטור המאפיינים של מילה במסמך כולל: 1) קבוצת מספרים המיצגים את המילה; 2) קבוצת מספרים המיצגים את הקטגוריה שלה (שם עצם, פועל); 3) קבוצת מספרים המיצגים את התפקיד התחבירי שלה (נושא, נשוא, מושא).

המחשת הייתרונות של הייצוג הווקטורי של כל מאפיין: Word Embedding
מודל זה מייצג כל מילה בקורסוס הנתון ע"י ווקטור שהערכים נגזרים על פי המופעים השונים של המילה בקורסוס, בהקשרים שונים, ובפרט על פי היחסים השונים של המילה עם מילים אחרות.
במאמר המפורטים שהציג מודל זה, הומחשה עצמתו של הייצוג הווקטורי החדש של המילים (לא-CSIMIN)
בשפה, 'מילה', אלא כווקטור שערכי נלמדו מתוך הקורפוס) על מציאת קשרים 'מדמיימים' בין הייצוג הווקטורי של מילים בעלות קשר סמנטי:
מלך – זכר + נקבה ~ = מלכה
לונדון – אנגליה + צרפת ~ = פריז

פונקציות: במקום פונקציה אחת, ניתן להגדיר רשות של פונקציות המאורגנות בשכבות



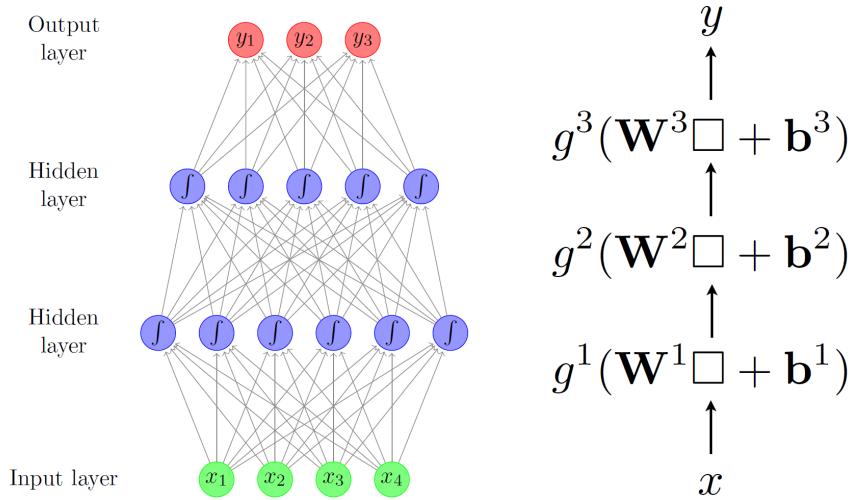
קביעת הארכיטקטורה של הרשת

- כמה שכבות ברשת
- כמה פונקציות בכל שכבה
- لأن הולך כל פלט של פונקציה בשכבה שמעליה
- מה כל אחת מהפונקציות
- מה ה'מקדמים' של כל פונקציה

ומובן...

יש ארכיטקטורות 'מקובלות' (בעקבות ניסוי וטעיה). לדוגמה:

- פונקציה אחת בכל שכבה
- הפונקציה מהצורה $b + \mathbf{W}\mathbf{x}$ (f). כלומר בהינתן וקטור הקלט \mathbf{x} , מכפילים אותו במטריצה \mathbf{W} ומוסיפים את הווקטור \mathbf{b} ומבצעים על המתקבל את הפונקציה f (דומה למקרה הפרטי של הפונקציה הליארית הקלאסית $b + ax = f(x)$)
- פונקציות f (או g לצורך למטה מימין) שהתגלו כיעילות: sigmoid, tanh, hard tanh, rectified linear unit



3 g_{1,2,3} יבחרו מתוך סדרית הפונקציות המומלצות, ומטריצות ה W ווקטוריו ה b (הפרמטרים הקבועים של הפונקציות) ילמדו במהלך האימון.

ארQUITקטורה פשוטה זו מכונה MLP.
בארכיטקטורה זו, מלבד האימון האוטומטי על בסיס הדוגמאות, מעצב הרשות רק בוחר את מספר השכבות ואת סוג הפונקציה (ניסוי וטעיה).

לאחר שהרשת מאומנת, ככלור יש את המטריצות של הפרמטרים הקבועים של הפונקציות, ניתן להפעיל על אובייקטים חדשים:

- o ייצוג האובייקט כסדרה של ווקטורי מאפיינים
- o הפעלת הרשת על הקלט
- o הפלט של כל פונקציה מועבר להלאה, עד השכבה הסופית
- o המרת הייצוג הווקטורי של הפלט למשגיא הבעיה

לדוגמה, עבור סיווג המסמכים:

- o ייצוג המסマー כסדרה של ווקטורי מאפייני המיללים
- o הפעלת הרשת על הווקטורים המציגים את המסマー
- o פלט: ווקטור אחד (שגודלו כמספר הנושאים) המגדיר את התפלגות הנושאים עבור מסマー זה (כנישה 3 בווקטור מתארת את מידת השיכנות של המסマー בקהלט לנושא מס' 3).

בעיה: בארכיטקטורה הפשוטה של MLP יש קלט עם ווקטור אחד בלבד. אך אם היי ברשות מספר ווקטורים, המספר חייב להיות קבוע מראש (לא בונים רשת שונה לכל מסマー, אלא רשת גנרטית עבור סיווג מסוימים באשר הם), בעוד שבמקרה שלנו מסマー מיוצג על ידי מערך של ווקטורים בגודל משתנה (כמספר המיללים)

נמצג ייחדי את כל מערך וקטורי המאפיינים לווקטור אחד: שרשור אחד אחריו השני, או סכימה שלהם, למשל חישוב הממוצע של כל קורדינטה (זה חזוי, אך זה עובד). הסכימה של הווקטור מכונה WOB (Bag Of Words). היא מתוארת בנוסחה הבאה (סכום הווקטורים $f_1 \dots f_k$ לווקטור אחד):

$$CBOW(f_1, \dots, f_k) = \frac{1}{k} \sum_{i=1}^k f_i$$

בעיה: במקרה שלנו (סיווג מסוימים), הפלט הוא התפלגות הנושאים, ההסתברות שהמסגר שייר לכל אחד מהנושאים, בעוד השרשת מוציאה וקטור שאינו בהכרח התפלגות.

נחותר את המידע בווקטור לתפלגות. שיטה מקובלת: softmax. הנוסחה הבאה מגדירה כיצד הופכים באופן זה וקטור כללי x לווקטור המיצג התפלגות:

$$x = x_1, \dots, x_k \text{ softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}}$$

אימון

נקודות מזע: ארכיטקטורה מסוימת שנבחרה בנייסוי וטעיה: כמה שכבות, מה הפונקציה.
בහינתן אוסף גדול של זוגות קלט-פלט (מסגר ותמהיל נושאים), תהליכי האימון לומדים:
 - כיצד לייצג וקטורי טוב יותר את הקלט והפלט
 - מהם הפרמטרים של הפונקציות ברשות (מטריצות W וקטורי b במקרה שלנו)

תבנית כללית:

- עברו כל איטרציה אימון
- o עברו כל דוגמא (מסגר ותמהיל נושאים)
 - יציג הקלט באופן וקטורי (ממוצע וקטורי המילים במסמך ע"פ WOB)
 - הפעלת הרשות על וקטור הקלט [forward]
 - השוואת פלט הרשות (ווקטור הנושאים, לאחר המרת הפלט לווקטור התפלגות ע"י softmax) לפלט הנוכחי בדוגמה (תמהיל הנושאים 'הנכון') וחישוב הפרער ביןיהם [loss]
 - פונקציות loss מקובלות: hinge, log, cross entropy, ranking
 - עדכן יציג הקלט והפרמטרים של הפונקציות בהתאם לפער זה [update]
 - פונקציות תיקון מקובלות: Stochastic Gradient, Nesterov Momentum

בדוגמה, מפעלים על loss את הגרדיאנט של הפונקציה הנוכחית, כלומר על וקטור הנגזרות החלקיים (כל פעם על פרמטר אחר)

מאחר שמדובר במטריצות גדולות, עבור מאפיינים רבים (לעתים מיליards), בניית הגרדיינט לאחר כל דוגמא כבده ביוטר. אפשרויות מקובלות:

Batch normalization -
במקום להזין דוגמא אחת כל פעם לרשף, מזינים אחד קבוצה של דוגמאות, כלומר מטריצה במקום ווקטור. לשם כך יש להוסיף מידע למטריצה A ולווקטור B של כל פונקציה. מעבד GPU מקבלים ברמת החומרה את החישוב של טנסוריים שכאה.

עיבוד מקביל של קבוצות דוגמאות שונות -
כאשר המטריצות גדולות מאוד, נרמול הבاطן אינו מספיק. במקרה זה, נפצל את אוסף הדוגמאות לקבוצות אימון שונות, נריץ כל קבוצה על מחשב נפרד (Map), ונמזג את כל התיקונים בסוף כל איטרציה (Reduce):

[\[Chu et al., Map-Reduce for ML on multi-core, NIPS 2006\]](#)

Class Mapper

```
method Initialize
    nn := load prev model

method Map(exampleId, <x,y>)
    y' := forward(x)
    ls := loss(y,y')
    backward(ls)

method Close()
    for parameter in nn.parameters // for each matrix  $W_i$  or vector  $b_i$ 
        Emit(parameter.id,parameter)
```

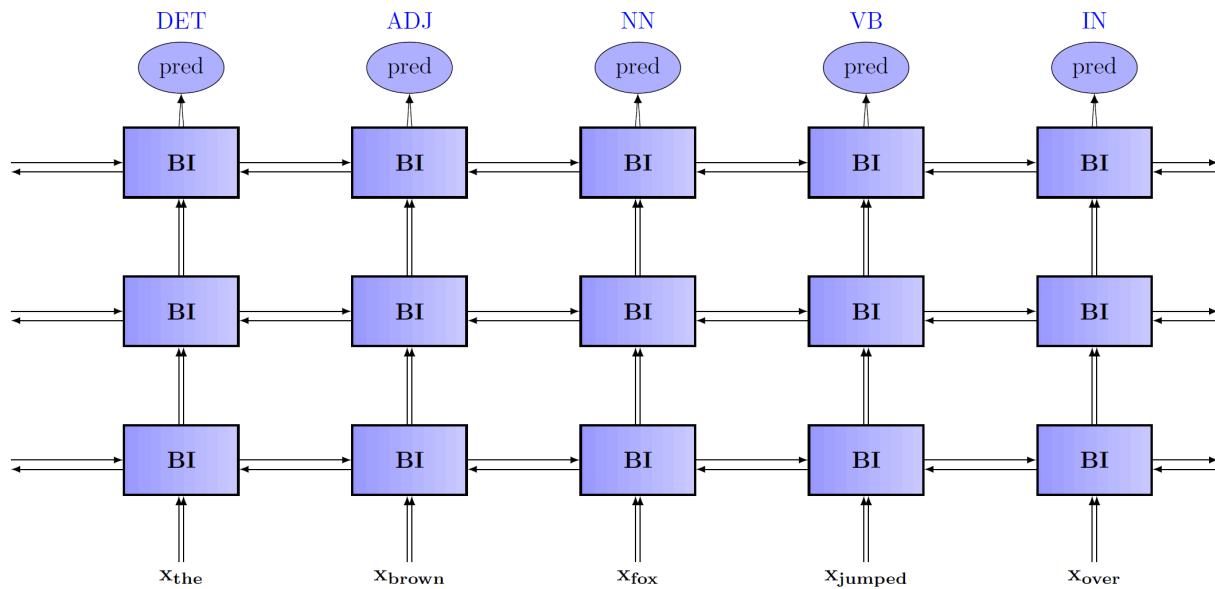
class Reducer

```
method Reduce(parameterId, values)
    sum := new Matrix // with one or more columns
    size = 0
    for value in values
        sum  $\oplus_{add}$  value // add each cell of matrix value to the corresponding cell in sum
        size := size + 1
    newParameter = sum / size // div the value of each cell in sum by size
    Emit(parameterId, newParameter)
```

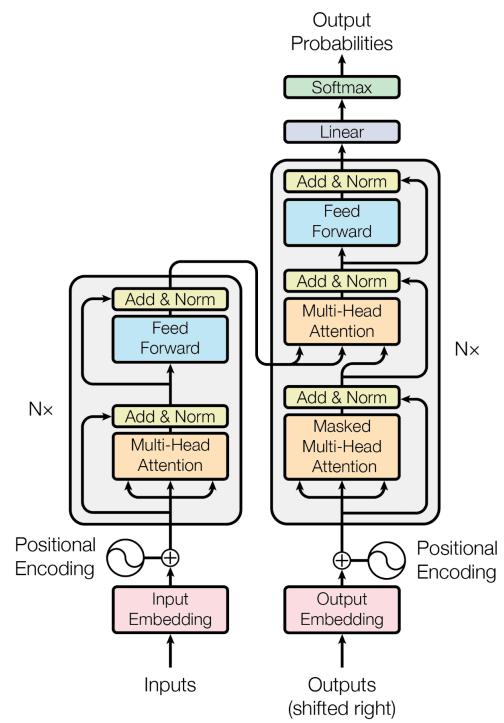
ניתן כמובן להגדיר ארכיטקטורות מורכבות יותר מ-MLP, וכך אכן עושים (והרבה). עבור בעיית התיאוג (מציאת הקטגוריות של מילים בטקסט), בה עסקנו בהרחבה, לדוגמה, ניתן להשתמש

בארQUITטורה המכונה LSTM. בארQUITטורה זו, כל רכיב (שהוא MLP בעצמו) משפיע על הרכיב המקביל הקודם, על הרכיב המקביל הבא, על השכבה מתחתיו, ועל השכבה מעליו.

כך נראה הרשות בארQUITטורה זו עבור עשיית התיאוג:



כiom מקובל מאד לעבוד עם ארQUITטורת transformer (הכוללת attention):



9. הוספת שכבת caching ל Hadoop

מוטיבציה

ראינו מספר אלגוריתמי למידה (CMDOMNI שבעה). המשותף לכל האלגוריתמים היה **המעבר החזר ונשנה על הקלט**, לשם למידה ושיפור המודל. בעיצוב ה-R-M של האלגוריתמים, ה Mappers בכל איטרציה קיבלו את אותו קלט מהאיטרציה הקודמת, אך לא הייתה בהכרח השמה של אותו split לאותו mapper. ניתן לעיל את קריאת הקלט (הקורפוס המקורי), על ידי מנגנון שייתן לכל Mapper, בשאייפה, splits שהוא קרא כבר באיטרציות קודמות. באופן זה, ניתן לחסוך את קריאת הקלט מערכות הקבצים המבוזרת, ולקרוא אותו מקומית מהDISK או מהזיכרון.

עיצוב

- אחסון מקומי של splits (בזיכרון או בDISK) שימוש חוזר ב split מקומי
 - o נגיד רשיון גרסאות של RecordReader גנרי:
 - o CacheLoaderRecordReader
 - o RecordReader זה מיועד לפעם הראשונה שבה ה split נשמר. כל $\langle V, K \rangle$ שהRecordReader קורא (במתודת nextKeyValue) יישמר בנוסף לוקאלית על המחשב הנוכחי, בDISK או בזיכרון, במימוש שלנו הוא יישמר בזיכרון משותף (לא סתם בHeap של התהילה של ה Mapper, כך שהוא יהיה נגיש באיטרציות הבאות לכל Task Tracker שraz על מחשב זה).
 - o CacheRecordReader
 - o RecordReader זה מיועד ל'פעמים הבאות' שבהם נדרש לקרוא את ה split (באיטרציות הבאות של אלגוריתם האימון) – הקריאה (במתודת nextKeyValue) לא תבצע מערכות הקבצים המבוזרת אלא מהמידע שנשמר לוקאלית (בזיכרון המשותף, במימוש שלנו)
 - o שני ה RecordReader ימומשו באופן גנרי, ככלומר הם לא קובעים כיצד מנתחים את ה split כרשימת $\langle V, K \rangle$ (זה יעשה ע"י RecordReader פנימי שיינטן כפרמטר).
- ה InputFormat יקבע איזו RecordReader מתאים (במתודה createRecordReader), לאור המידע האם ה split קיים כבר מקומית במחשב של ה Task Tracker הנתון.
- ניהול מבנה נתונים ברמת הג'וב, המפה Task Tracker לרשימת splits המאוחסנים לוקאלית במחשב שלו.
- עדכון הקוד בסביבת Hadoop איזה split לחתול Task Tracker פניו. ככלומר, הוספה השיקול של איזה splits מאוחסנים לוקאלית במחשב של ה Task Tracker הפנו (על פי מבנה הנתונים הנ"ל) למערכת שיקולי ההשמה.

מימוש

```
class CacheInputFormat<K extends Writable, V extends Writable> extends InputFormat<K, V> {  
    ...  
    private InputFormat<K, V> delegateInputFormat = null;  
    private NodeCacheManager cacheManager;
```

```

...
public RecordReader<K, V> createRecordReader
    (InputSplit split, TaskAttemptContext context) throws IOException, InterruptedException {
    ...
    // If the split was already cached
    if (cacheManager.contains(splitId)) {
        CacheRecordReader<K, V> cacheRecordReader = new CacheRecordReader<K, V>();
        cacheRecordReader.initialize(split, context);
        return cacheRecordReader;
    } else {
        CacheLoaderRecordReader<K, V> cacheLoaderRecordReader =
            new CacheLoaderRecordReader<K, V>();
        cacheLoaderRecordReader.setDelegateRecordReader(
            this.delegateInputFormat.createRecordReader(split, context));
        cacheLoaderRecordReader.initialize(split, context);
        return cacheLoaderRecordReader;
    }
}
}

class CacheLoaderRecordReader<KEYIN extends Writable, VALUEIN extends Writable> extends
RecordReader<KEYIN, VALUEIN> {

    private RecordReader<KEYIN, VALUEIN> delegateRecordReader = null;
    private SharedMemory sm;
    private AdjustableDataOutput preliminaryOutpt;
    ...
    public boolean nextKeyValue() throws IOException, InterruptedException {
        boolean delegateNext = this.delegateRecordReader.nextKeyValue();
        /* If there is something to read, we need to remember to store it in the cache */
        if (delegateNext) {
            curKey = this.delegateRecordReader.getCurrentKey();
            curValue = this.delegateRecordReader.getCurrentValue();
            // Read the data into an adjustable data output
            curKey.write(preliminaryOutpt);
            curValue.write(preliminaryOutpt);
            ...
        }
        return delegateNext;
    }
}

```

```

public void close() throws IOException {
    this.delegateRecordReader.close();
    // If no records are stored, just enter a dummy entry in the cache table
    if (recordsStored == 0) {
        CacheDataEntry entry =
            new CacheDataEntry(cacheLocation, 0, recordsStored, "", "");
        // Add the entry to the node's cache manager
        NodeCacheManager.INSTANCE.add(splitId, entry);
        ...
        sm = new SharedMemory(cacheLocation, preliminaryOutput.getSize(), true);
        ...
    }
}
}

```

```

class CacheRecordReader<KEYIN extends Writable, VALUEIN extends Writable> extends
RecordReader<KEYIN, VALUEIN> {

    private SharedMemory sm;
    private DataInput cacheInput;

    public void initialize(InputSplit split, TaskAttemptContext context)
        throws IOException, InterruptedException {
        String cacheld = conf.get(CacheInputFormat.DELAGATE_INPUT_FORMAT_ID);
        CacheInputFormatId splitId = new CacheInputFormatId(split, cacheld);
        CacheDataEntry entry = NodeCacheManager.INSTANCE.get(splitId);
        ...
        sm = new SharedMemory(entry.getLocation().toString(), entry.getSize().get(), false);
        ...
        cacheInput = new CacheBufferedDataInput(sm);
        ...
    }

    public boolean nextKeyValue() throws IOException, InterruptedException {
        boolean hasNext = (recordsNum - recordsRead) > 0;
        if (hasNext) {
            curKey.readFields(cacheInput);
            curValue.readFields(cacheInput);
            recordsRead++;
        }
        return hasNext;
    }
}

```

```
}
```

בגרסת הקודמת של Hadoop, המחלקה JobInProgress יציג Job שרצ, וכן המחלקה TaskInProgress יציג TaskTrackerStatus שרצ, ו TaskTracker יציג מסימה לביצוע. במחלקה ה JobInProgress קיימת מתודת הבוחרת איזה split לחת לעיבוד במשימת Mapper עבור TaskTracker פנו. במתודה זאת ממומנת למשה מדיניות הקצאה של הסביבה, בפרט העדפה לחת ל TaskTracker ספליט ה'קרוב' אליו מבחינת תקשורת.

נעדן מתודה זו, כך שתקדם כל תבזוק האם אחד הספליטים האפשריים עבור ה Task ה פנו הנתון מואחסן בזיכרון המשותף של המחשב שלו, עקב מעבר קודם על ספליט זה ע"י TaskTracker כleshon במחשב זה.

```
class JobInProgress {  
...  
    private synchronized int findNewMapTask(TaskTrackerStatus tts,...) {  
        ...  
        // @HC-BEGIN  
        tip = findCachedFromList(cacheForLevel, tts);  
        if (tip != null) {  
            scheduleMap(tip);  
            return tip.getIdWithinJob();  
        }  
        ...  
    }  
}  
  
private TaskInProgress findCachedFromList(List<TaskInProgress> tips, TaskTrackerStatus tts) {  
    ...  
    return CacheState.INSTANCE.getBestTip(conf, tips, tts.getHost());  
}
```

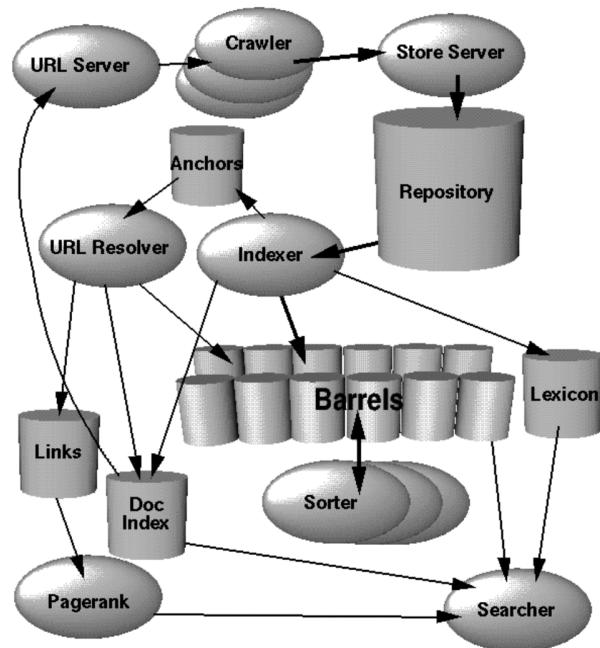
אם רוצים לעבוד עם מגנון ה Caching יש לציין בהגדרת ה Job:

```
job.setInputFormatClass(DBInputFormat.class);  
...  
job.setInputFormatClass(CacheInputFormat.class);  
CacheInputFormat.setDelegateInputFormatData(job,  
    DBInputFormat.class, connString + " | " + queryString);
```

בבדיקה ניסויית התגלה כי המעבר הראשון על הקורפו דרש 10% זמן יותר (בשל ניהול השמירה של כל >V,K< שנקרו או מה Split לזכרו cache), אך המעברים הבאים חסכו 90% מהזמן.

10. מנוע חיפוש

10.1 מבוא – הארכיטקטורה של מנוע החיפוש גוגל



10.2 מפתח (Indexing)

ה indexer בונה ממגר קבצי הרשות, מבנה נתונים המפה מילה לרשימות הקבצים שבhem היא מופיעה. נשים לב לכך, שפעולות indexer למעשה הופכת מיפוי של קובץ π מילים, למ בניית מילה π קבצים. שחולוף שכזה הינו קלости לתוכנית Map-Reduce.

Class Mapper

method Map(docId, doc)

```
words := new Set  
for w in doc  
    words.add(w)
```

For w : words

Emit(w,docId)

Class Reducer

```
method Reduce(word, docIds)
    for docId in docIds
        Emit(word, docId)
```

ביצוג זה, המידע עבור כל מילה הוא רק מספר הקובץ שבו היא מופיעה. אפשר כמובן להרחב מידע זה, בפרט אפשר להעшир אותו במאפיינים שיעזרו לדרג את תוצאות החיפוש:

- היקן המילה מופיעה במסמך
- האם המילה מופיעה בכותרת, בתיאור של תמונה, בטקסט של קישור, ועוד

Class Mapper

```
method Map(docId, doc)
    word2properties := new Table
    for w in doc
        updateWordProperties(w, getWordDocProperties(), word2properties)
    for <w, props> : word2properties
        Emit(w, docId, props)
```

Class Reducer

```
method Reduce(word, values)
    for <docId, wordDocProperty> in values // sorted by wordDocProperties!
        Emit(word, docId)
```

10.3 חיפוש

בהינתן שאלתא, המורכבת ממספר מילות חיפוש, ניתן לחלץ את רשימת הקבצים בהם מילים אלו מופיעות בעזרת lookup פשוט על המפתח שנוצר בשלב האינדוקס (לעיל).

השאלה המרכזית היא: יכיז דרגת תוצאות אלן, כך שהמשמעותים הרלוונטיים ביותר יופיעו בהתחלה?

קטגורית, קיימות שתי גישות מרכזיות: דירוג מונחה שאלתא, דירוג מונחה מסמך

דירוג מונחה שאלתא

הדירוג של כל מסמך מבוסס על מידת ההתאמה בין השאלתא למסמך. כלומר, תוכן המסמך הći מתאים לשאלתא.

אחד המՃדים הבסיסיים והקלואסיים עבור תאיות תוכן זו היא tf-idf.

מדד זה מבוסס על ההנחה האינטואיטיבית והפשוטה, כי מסמך שמכיל פעמים רבות את מילות החיפוש בשאלתא תואם לשאלתה.

Term Frequency – השכיחות היחסית של מילה במסמך j : מספר הפעמים בהם מופיעה המילה j במסמך j, חלקו מספר המילים במסמך.

$$tf_{i,j} = n_{i,j} / |d_j|$$

בעיה: לעיתים, שכיחותה היחסית של מילה במסמך לא מלמדת על מידת ההתאמה של השאלה למסמך. לדוגמה, השאלה 'את חפירה'. המילה 'את' מופיעה הרבה בכל מסמך, אך שפער השכיחות בין מסמכים המתיחסים ל'את' כשם עצם ובין אלה שלא אינם גדולים כל כך.
גנסה להקטין את המשקל של מילים אלו, ע"י הוספת רכיב 'mobekot' למדד.

– עד כמה מילה מסוימת שכיחה בכלל המסמכים במאגר. Document Frequency

$$df_i = | \{ d : t_i \text{ in } d \} | / |D|$$

כל שמליה נדירה יותר (מופיעה במעט מסמכים), רמת המובוקות שלה גדולה יותר (כלומר, נרצה לחזק מאוד מסמכים המכילים אותה בכל זאת).

כל שהמליה שכיחה יותר (מופיעה כמעט בכל מסמך), רמת המובוקות קטנה יותר (כלומר, לא ניקח ברצינות את העובדה שהיא מופיעה במסמך מסוים).

בעיה טכנית: ערך ה df גבוה כאשר רמת המובוקות נמוכה, ולהפך. יוצר בעיה בשילוב המדדים – נהפוך אותו טכנית:

$$idf_i = |D| / | \{ d : t_i \text{ in } d \} |$$

המדד המשקול $tf=idf$

$$Tf-idf_{i,j} = tf_{i,j} \cdot idf_i$$

ניתן לחשב, בעיבוד מקדים, את הערך $Tf-idf_{i,j}$ עבור כל מילה i במאגר ועבור כל מסמך j במאגר – משימה כללואית ל-R-M (דומה לתרגיל 2)

באופן זה, כל מסמך מיוצג ע"י וקטור המציין בכל קורדינטה את מידת ההתאמה של כל אחת מהמלים במאגר למסמך זה.

בහינתן שאלתא, נבנה ווקטור דומה עבור מסמך המכיל את המילים בשאלתא (רבו המוחלט יהיה 0), ונמדד את המרחק בין הווקטור המציג את השאלה ובין הווקטורים המייצגים את המסמכים השונים, ונדריך על פי מרחק זה (קיימות מגוון גישות למדד מרחק בין וקטורים)

[עבור שאלות עם מילה בודדת, ניתן מראש לכלולמבנה `wordDocProperty` שהאינדקסר בונה עבור כל מילה בכל מסמך, את ציון ה $tf-idf$ של המילה במסמך זה, וכלול אותו בקריטריון המיוון (לצד מידע כמו

אם היא מופיע בכוורתה וכו') של המסמכים במתודת ה `reduce` של האינדקסר, כך שהם ישמרו מראש בפתח על פי מידת התאמתם למילה]

דירוג מונחה מסמן

בגישה זו, אנו מדרגים מראש את כל המסמכים על פי מידת חשיבותם במאגר, ללא קשר לשאלתא מסוימת.

כיצד נקבע האם מסמן הוא חשוב, ועד כמה? פיג' ובירן: מסמן חשוב הוא מסמן שהסתברות להגיע אליו בשיטוט אקראי בראשת גבואה. בפרט, מסמן חשוב הוא מסמן שיש אליו קישורים ממומכים חשובים אחרים.

כיצד מגאים לדף בראשת ח בשיטוט אקראי?

- ע"י בחירת דף זה מבין כל G הדפים בראשת (בהתברות של $1 / |G|$)
- דרך קישור מדף אחר w שאנו נמצאים בו כתע (בהתברות הימצא בדף w , חלק מס' w הקישורים לדפים אחרים מדף w)

$$P(n) = \alpha \left(\frac{1}{|G|} \right) + (1 - \alpha) \sum_{m \in L(n)} \frac{P(m)}{C(m)}$$

כאשר:

- א) קבוצת הדפים הכוללת קישור לדף w
- ב) מספר הקישורים הכולל בדף w

כדי לאזן ולשלוט על מידת הרסתמוכות על הקритריון הראשון (בחירה מוגנת של הדף) ועל הקритריון השני (הגעה מדף אחר), נגדיר פרמטר `alpha` בין 0 ל 1 (ברירת המחדל בזמןנו של גוגל הייתה 0.2) `alpha = 0.2`

ונתמקד במימוש הקритריון השני ($0 = alpha$)

עיצוב ב-R-M

הרעין הכללי:

- קיימ מאגר ביג-דאטה המתאר את הדפים השונים ואת הקשר ביניהם, באופן הבא:
`pageId [out pages ids] rank`

- מתודת `map` מקבלת תיאור של אחד הקודקודים, ושולחת את הדרגה הנוכחית שלו לקודקודיו השכנים (ביחס למספרם, כולם נחלק את הדרגה במספרם).
- מתודת `reduce` מקבלת עבור קודקוד את הדרגות של כל הקודקודים המובילים אליו, ומחשבת את הדרגה שלו מחדש.
- כל סיבוב של R-M הוא איטרציה אחת של חישוב הדרגה ממשיכים, עד שהדרגות לא משתנות

Class Mapper

```
method Map(pageId, page)
    for pid in page.out
        Emit(pid, page.rank / | page.out |)
```

Class Reducer

```
method Reduce(pageId, values)
    newRank := 0
    for (value in values)
        newRank := rank + value
    Emit(pageId, newRank)
```

בעיה טכנית: בתום סיבוב ה-R-M איבדנו את נתונים הקודקודים ונשארנו רק עם הדאטה שלהם – הדרגה.

בעוד שבסיבוב הבא נדרש מידע זה (בפרט מי הקודקודים היוצאים ממנו)

- ניתן לבצע join תיאור הקודקודים עם הדרגה החדשה, כפולה מקדים לסיבוב הבא.

- נבחר בדרך נוחה יותר, נדאג לכך שהMapper שולח גם את הגדרת הקודקוד עצמו לReducer

Class Mapper

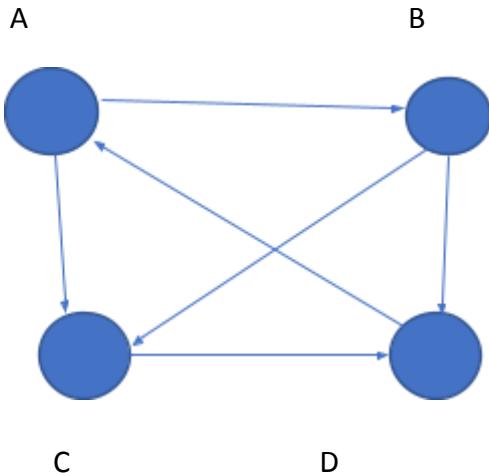
```
method Map(pageId, page)
    Emit (pageId, page)
    for pid in page.out
        Emit(pid, page.rank / | page.out |)
```

Class Reducer

```
method Reduce(pageId, values)
    myPage := null
    newRank := 0
    for (value in values)
        if (isPage(value))
            myPage := value
        else
            newRank := newRank + value
```

```
myPage.rank = newRank
Emit(pageId, myPage)
```

דוגמא:



$$P(A) = P(B) = P(C) = P(D) = 0.25$$

Map

```

<A, <A, [B,C], 0.25>> ↗ <A, <A, [B,C], 0.25>> >B, 0.125 < <C, 0.125>
<B, <B, [C,D], 0.25>> ↗ <B, <B, [C,D], 0.25>> <C, 0.125> <D, 0.125>
<C, <C, [D], 0.25>> ↗ <C, <C, [D], 0.25>> <D, 0.25>
<D, [A], 0.25>> ↗ <D, <D, [A], 0.25>> <A, 0.25>

```

Reduce

```

A [0.25, <A, [B,C], 0.25>] ↗ <A, <A, [B,C], 0.25>>
B [0.125, <B, [C,D], 0.25>] ↗ <B, <B, [C,D], 0.125>>
C [0.125, 0.125, <C, [D], 0.25>] ↗ <C, <C, [D], 0.25>>
D [0.125, 0.25, <D, [A], 0.25>] ↗ <D, <D, [A], 0.375>>

```

בניסוי הראשון של גугл (322 מיליון מסמכים), האלגוריתם התתכוון פחות או יותר אחרי 52 איטרציות.

נשים לב, כי מדובר בבתבנית כללית של עיבוד גרף מבוזר:

Initialize nodes data _____

Class Mapper

```

method Map(nodId, node)
    Emit (nodId, node)

```

```
for (nid in node.out)
    Emit(nid, node.data _____)
```

Class Reducer

```
method Reduce(nodeId, values)
    myNode := null
    newData := _____ // initialization
    for (value in values)
        if (isNode(value))
            myNode := value
        else
            newData := newData _____ value
    myNode.data = newData _____
    Emit(nodeId, myNode)
```

עבור האלגוריתם שלנו לדירוג הדפים:

Initialize nodes $1/|G|$ for each of the G nodes

Class Mapper

```
method Map(nodeId, node)
    Emit (nodeId, node)
    for (nid in node.out)
        Emit(nid, node.data / |node.out|)
```

Class Reducer

```
method Reduce(nodeId, values)
    myNode := null
    newData := 0 // initialization
    for (value in values)
        if (isNode(value))
            myNode := value
        else
            newData := newData + value
```

```
myNode.data = newData  
Emit(nodId, myNode)
```

נשתמש לדוגמה בתבנית זו, כדי לחשב את המרחק של כל הקודקודים מקודקוד נתון (אלגוריתם דיאקسطה הסדרתי)

נתון גרפ' מבוצר של קודקודים, כאשר השדה data מייצג את המרחק של כל קודקוד מהקודקוד מסוים. בהתחלה, הערך data של הקודקוד המסוים הוא 0, ושל כל השאר אינסוף.

Initialize nodes data **0** for the target node, **Infinity** for each other

Class Mapper

```
method Map(nodId, node)  
    Emit (nodId, node)  
    for (nid in node.out)  
        Emit(nid, node.data + 1)
```

Class Reducer

```
method Reduce(nodId, values)  
    myNode := null  
    newData := infinity  
    for (value in values)  
        if (isNode(value))  
            myNode := value  
        else  
            newData := Min(newData, value)  
    myNode.data = Min(newData ,myNode.data)  
    Emit(nodId, myNode)
```