Μοντέλα Ανάκτησης Πληροφορίας

Μοντέλα IR (1/3)

- Εκτός από την αναπαράσταση των εγγράφων και τη δημιουργία των ευρετηρίων, το άλλο μέρος του IR είναι ο καθορισμός του βαθμού συσχέτισης του ερωτήματος με τα έγγραφα
- Η διαδικασία ονομάζεται **matching process**

- Η διαδικασία εξάγει μια ταξινομημένη λίστα των εγγράφων ως προς το βαθμό συσχέτισης
- Οι ranking αλγόριθμοι υιοθετούν διάφορες τεχνικές και πληροφορίες όπως την κατανομή της συχνότητας των στοιχείων ή στο Web το social aspect των σελίδων
- Το social aspect των σελίδων καθορίζεται από τα links που δείχνουν πάνω στη σελίδα

Μοντέλα IR (2/3)

- Τα πιο συνηθισμένα μοντέλα IR είναι τα ακόλουθα:
  - The Boolean Model
  - The vector space model
  - The probabilistic model

Μοντέλα IR (3/3)

- Γενικά, ένα IR μοντέλο είναι ένας αλγόριθμος, ο οποίος:
  - Δέχεται ένα ερώτημα q και ένα σύνολο εγγράφων $D=\{d_1, d_2, \ldots, d_N\}$
  - Αποδίδει ένα **συντελεστή ομοιότητας (similarity coefficient)** σε σχέση με το q σε κάθε έγγραφο $SC(q,d_i)$
  - Η τιμή αυτή μπορεί να απαντηθεί και ως **τιμή κατάστασης της ανάκτησης (retrieval status value)**

The Boolean Model (1/12)

- Πρόκειται για ένα απλό μοντέλο που βασίζεται στη θεωρία συνόλων και στη Boolean άλγεβρα
- Τα ερωτήματα αναπαριστώνται ως Boolean εκφράσεις σε σχέση με τα στοιχεία
- Μπορούν να χρησιμοποιηθούν Boolean τελεστές AND, OR, NOT
- Παράδειγμα: photo AND mountain OR snow
- Η σημασία ενός στοιχείου t_i αναπαριστάται με δυαδικά βάρη
- Ένα βάρος $w_{ij} = \{0,1\}$ σχετίζεται με το tuple (t_i, d_j) ως μια συνάρτηση $R(d_j)$, με το σύνολο των στοιχείων του ευρετηρίου, την $R(t_i)$ και το σύνολο των εγγράφων στα οποία το στοιχείο του ευρετηρίου εμφανίζεται

The Boolean Model (2/12)

- Η συσχέτιση σχέση με το ερώτημα μοντελοποιείται σαν μια δυαδικής μορφής ιδιότητα για κάθε έγγραφο, i.e., $SC(q,d_j)=0$, $SC(q,d_j)=1$

- Υιοθετούμε την προσέγγιση του κλειστού κόσμου (close world assumption) όπου η απουσία ενός στοιχείου t σε ένα έγγραφο d είναι ισοδύναμο με την παρουσία του ¬t (άρνησης)

The Boolean Model (3/12)

- Ένα Boolean ερώτημα q μπορεί να ξαναγραφτεί σε μια διαζευκτική (disjunctive) κανονική μορφή – διάζευξη συζευκτικτών προτάσεων
- Παράδειγμα:
  - Το ερώτημα $q = t_a \wedge (t_b \vee \neg t_c)$
  - Μπορεί να γραφτεί ως $q_{\text{DNF}} = (t_a \wedge t_b \wedge t_c) \vee (t_a \wedge t_b \wedge \neg t_c) \vee (t_a \wedge \neg t_b \wedge \neg t_c)$
  - Για αυτό το ερώτημα όλο το σύνολο των εγγράφων μπορεί να χωριστεί σε 8 κατηγορίες
  - Κάθε κατηγορία αναπαριστάται από μια συζευκτική πρόταση
  - Στη συνέχεια το ερώτημα μπορεί να ικανοποιηθεί όταν ένα έγγραφο ικανοποιεί τουλάχιστον μια επί μέρους πρόταση
  - Για το παραπάνω ερώτημα έχουμε:

    $q_{\text{DNF}} = (1, 1, 1) \vee (1, 1, 0) \vee (1, 0, 0)$

The Boolean Model (4/12)

- Έχουμε:

$$SC(q, d_j) = \begin{cases} 1 & \exists\, q_c | (q_c \in q_{DNF}) \wedge (\forall\, t_i \in q_c \rightarrow w_{ij} = 1) \\ & \quad \wedge (\forall\, t_i \notin q_c \rightarrow w_{ij} = 0) \\ 0 & \text{otherwise} \end{cases}$$

- Κάθε Boolean ερώτημα υπολογίζεται με την παραγωγή μιας λίστας εγγράφων που ικανοποιούν τις προαναφερόμενες προτάσεις

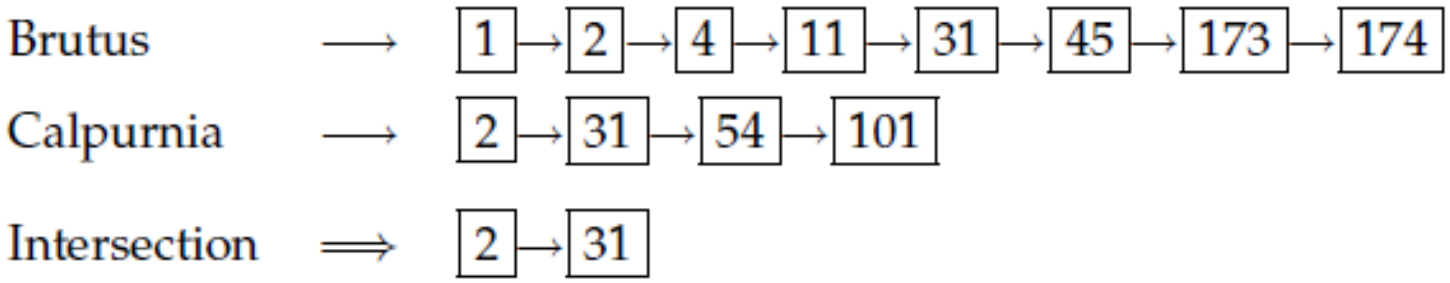
- Όταν πολλαπλές λίστες είναι διαθέσιμες για πολλαπλά ερωτήματα ή τμήματα ερωτημάτων τότε εφαρμόζουμε Boolean τελεστές για να παράξουμε το τελικό αποτέλεσμα

The Boolean Model (5/12)

- Παραδείγματα
  - q_1 OR q_2. χτίζουμε την ένωση (union) των λιστών των ερωτημάτων q1 και q2
  - q_1 AND q_2. απαιτεί την εξαγωγή της τομής των λιστών
  - q_1 AND NOT q_2. απαιτεί την εύρεση της διαφοράς των δύο λιστών

- Παράδειγμα
  - Ο υπολογισμός του αποτελέσματος t_a Λ t_b εμπλέκει τα ακόλουθα βήματα:
  - Εντοπισμός του t_a στο λεξικό
  - Ανάκτηση όλων των postings L_a
  - Εντοπισμός του t_b στο λεξικό
  - Ανάκτηση όλων των postings L_b
  - Υπολογισμός της τομής των L_a, L_b

The Boolean Model (6/12)



The Boolean Model (7/12)

- Στόχος είναι ο αποδοτικός υπολογισμός της τομής ώστε γρήγορα να βρούμε τα έγγραφα που περιλαμβάνουν και τους δύο όρους
- Διατηρούμε δύο δείκτες στις δύο λίστες
- Ελέγχουμε τα IDs των εγγράφων και στην ισότητα εισάγουμε το έγγραφο στην απάντηση
- Διαφορετικά προχωρούμε το δείκτη στη λίστα με το μικρότερο ID

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

The Boolean Model (8/12)

- Έστω το μήκος των λιστών είναι ίσο με M και N αντίστοιχα
- Ο υπολογισμός της τομής των λιστών απαιτεί O(M+N) λειτουργίες
- Γενικά, ο υπολογισμός της απάντησης για ένα ερώτημα απαιτεί Θ(K) όπου K είναι το πλήθος των εγγράφων

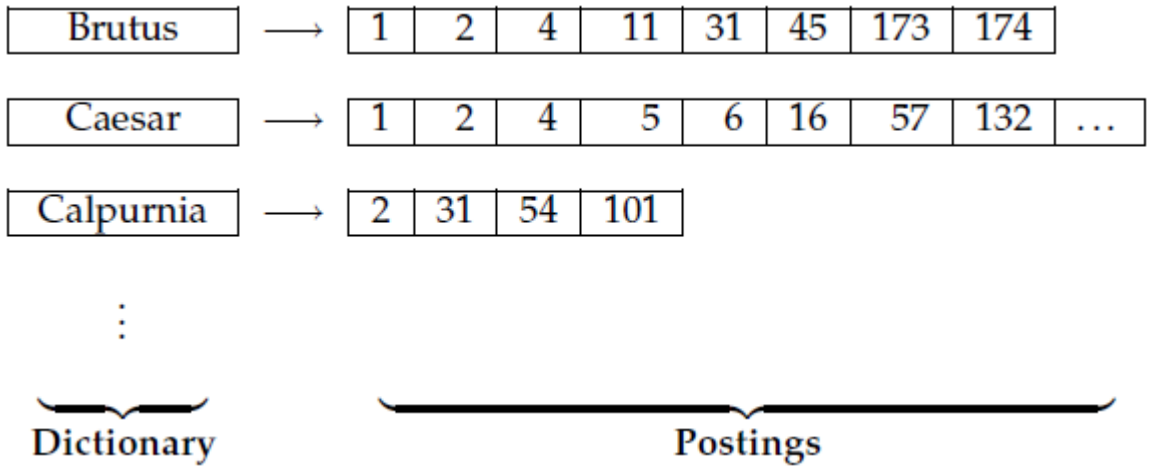
```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

The Boolean Model (9/12)

- Μπορούμε να επεκτείνουμε το μοντέλο για να διαχειριστούμε πιο πολύπλοκα ερωτήματα, π.χ. της μορφής (Brutus OR Caesar) AND NOT Calpurnia
- Η βελτιστοποίηση των ερωτημάτων (query optimization) έγκειται στην επιλογή της οργάνωσης της εξαγωγής των απαντήσεων
- Μια σημαντική εργασία είναι η σειρά με την οποία επεξεργαζόμαστε τα postings lists
- Αν εκκινήσουμε συγχωνεύοντας τις δύο πιο μικρές λίστες, τότε όλα τα ενδιάμεσα αποτελέσματα πρέπει να μην είναι πιο μεγάλα από τις μικρές λίστες

The Boolean Model (10/12)

- Για τις λίστες:



- Και το ερώτημα Brutus AND Caesar AND Calpurnia εκτελούμε το: (Calpurnia AND Brutus) AND Caesar

The Boolean Model (11/12)

- Σε πιο γενικά ερωτήματα όπως (madding OR crowd) AND (ignoble OR strife) AND (killed OR slain):
  - Παίρνουμε τις συχνότητες κάθε όρου
  - Υπολογίζουμε το μήκος κάθε OR αθροίζοντας τις συχνότητες κάθε τμήματος της πρότασης
  - Επεξεργαζόμαστε το ερώτημα σε αύξουσα σειρά μεγέθους κάθε όρου
- Υπολογίζουμε και αποθηκεύουμε τις απαντήσεις για ενδιάμεσες εκφράσεις που απαρτίζουν μια πιο πολύπλοκη έκφραση
- Πολλές φορές όμως τα ερωτήματα απαρτίζονται μόνο από συζεύξεις
- Σε αυτές τις περιπτώσεις συγχωνεύουμε κάθε λίστα με τα ενδιάμεσα αποτελέσματα
- Αρχικοποιούμε τα ενδιάμεσα αποτελέσματα με χρήση των λιγότερο συχνών όρων

The Boolean Model (12/12)

- Αλγόριθμος:

```
INTERSECT(⟨t_1, ..., t_n⟩)
1  terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
2  result ← postings(first(terms))
3  terms ← rest(terms)
4  while terms ≠ NIL and result ≠ NIL
5  do result ← INTERSECT(result, postings(first(terms)))
6     terms ← rest(terms)
7  return result
```

Dictionaries Management and Tolerant Retrieval

Δομές Αναζήτησης

- Πρώτα βήματα που κάνουμε όταν τίθεται ένα ερώτημα είναι:
  - Βλέπουμε αν ο κάθε όρος του ερωτήματος υπάρχει μέσα στο λεξικό
  - Ανακτούμε τους δείκτες προς τις postings για κάθε όρο
- Έχουμε στη διάθεσή μας δύο λύσεις για την ανάκτηση:
  - Κατακερματισμό (hashing)
  - Δένδρα αναζήτησης (search trees)

- Το τι θα επιλέξουμε εξαρτάται από τα ακόλουθα ερωτήματα:
  - Πόσα κλειδιά θα έχουμε;
  - Ο αριθμός τους θα είναι σταθερός ή θα μεταβληθεί σημαντικά;
  - Ποια είναι η συχνότητα με την οποία θα προσπελαύνουμε τα κλειδιά;

Δομές Αναζήτησης

- Κάποιες μηχανές αναζήτησης υιοθετούν hashing
- Κάθε όρος του λεξικού κατακερματίζεται σε ένα ακέραιο σε ένα μεγάλο εύρος επιλογών ώστε να αποκλείονται οι συγκρούσεις
- Αν υπάρξουν συγκρούσεις τότε επιλύονται με βοηθητικές δομές τις οποίες πρέπει να συντηρούμε

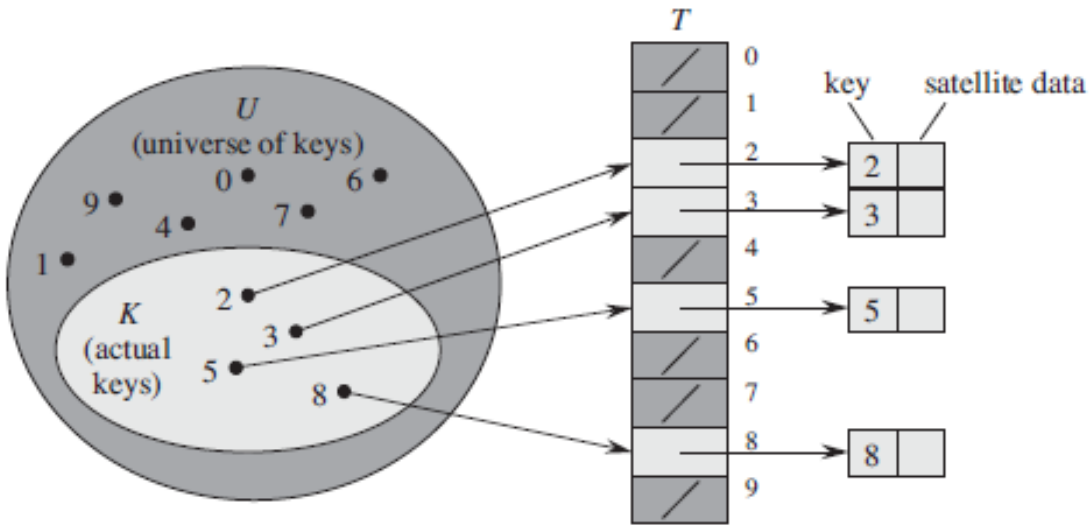
Δομές Αναζήτησης

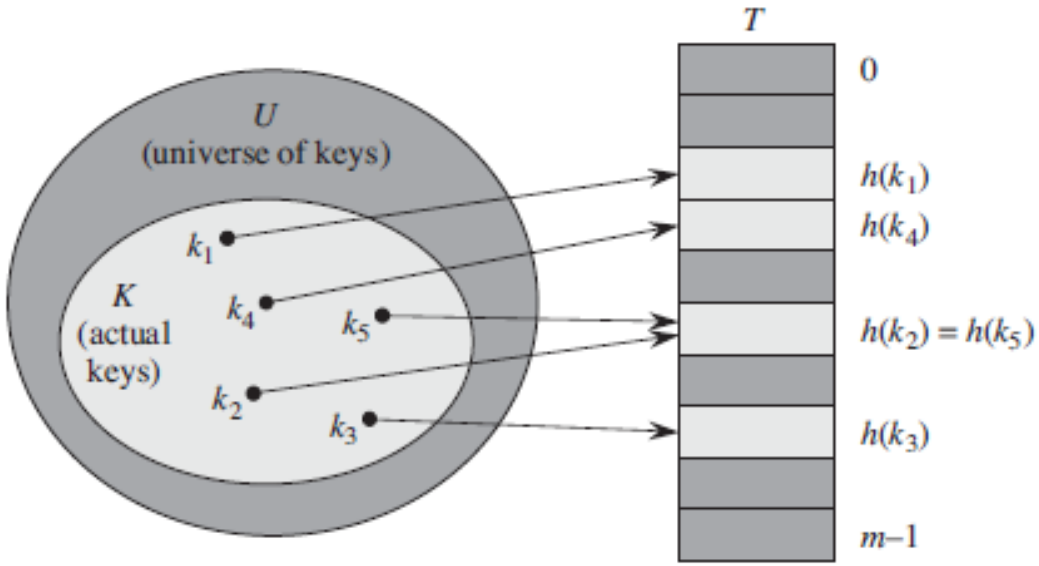
- Η έννοια του **κατακερματισμού (hashing)** βασίζεται στην ιδέα της διασποράς κλειδιών σε ένα μονοδιάστατο πίνακα που ονομάζεται **πίνακας κατακερματισμού (hash table)**
- Η κατανομή γίνεται μέσα από την εφαρμογή μιας συνάρτησης h που ονομάζεται **συνάρτηση κατακερματισμού (hash function)**

- Για κάθε κλειδί, η συνάρτηση αποδίδει ένα ακέραιο σε ένα διάστημα [0..m-1] που ονομάζεται **διεύθυνση κατακερματισμού (hash address)**
- Παράδειγμα: h(K)=K mod m

Δομές Αναζήτησης

- Παράδειγμα





Δομές Αναζήτησης

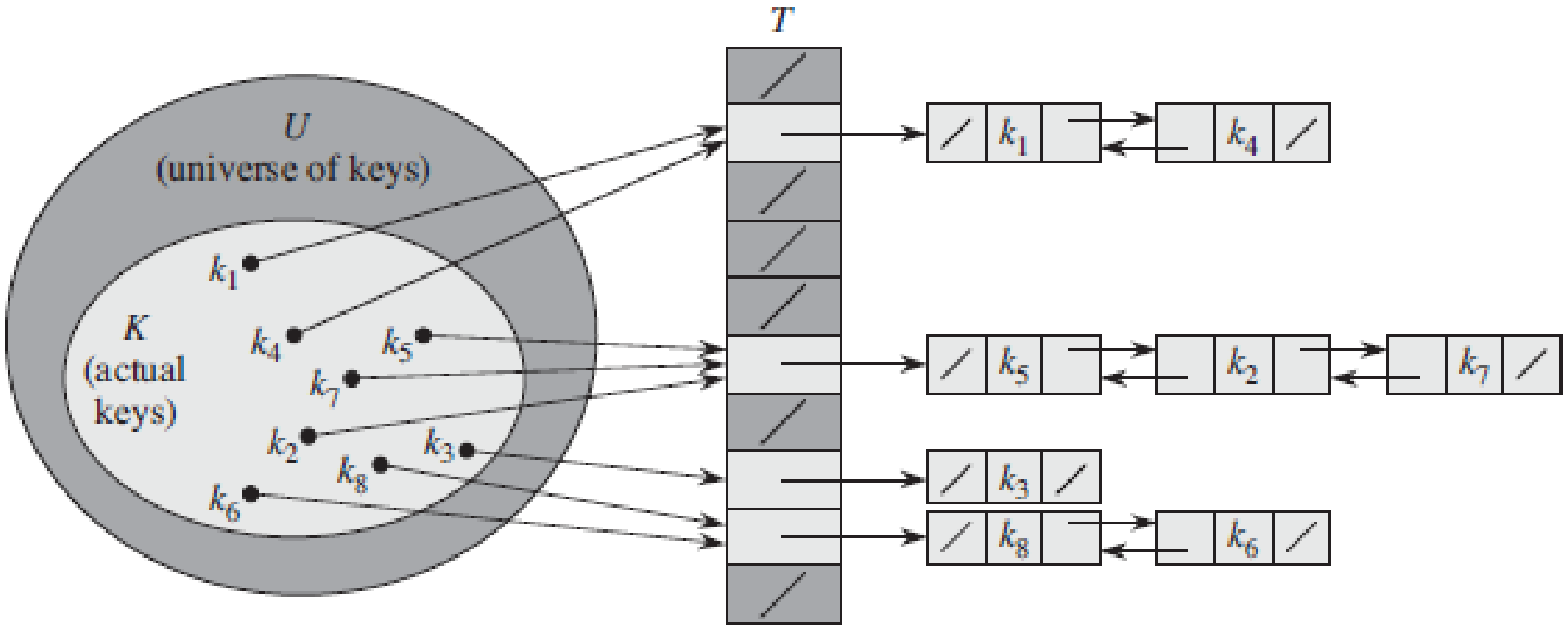
- Πρόβλημα
  - Δύο κλειδιά μπορεί να κατακερματιστούν στην ίδια θέση του πίνακα
    - **Σύγκρουση (collision)**
- Προσπαθούμε να βρούμε κατάλληλη και αποδοτική hash function
- Εφαρμόζουμε διάφορες αποδοτικές τεχνικές

Δομές Αναζήτησης

- Μια από τις διαθέσιμες λύσεις: chaining
- Τοποθετούμε τα κλειδιά που κατακερματίζονται στην ίδια θέση του πίνακα σε μια συνδεδεμένη λίστα
- Κάθε θέση του πίνακα περιέχει ένα δείκτη προς την κορυφή της λίστας

Δομές Αναζήτησης

- Παράδειγμα



Δομές Αναζήτησης

- Όταν τεθεί το ερώτημα κατακερματίζουμε κάθε όρο του ώστε να πάρουμε τους δείκτες προς τις λίστες

- Διαφορετικές εκφάνσεις ενός όρου (π.χ. Δημοκρατία, Δημοκρατικός) θα οδηγήσουν σε διαφορετική τιμή κατακερματισμού

- Δυστυχώς όσο μεγαλώνει ο όγκος του Web και των λεξικών αντίστοιχα, τόσο μεγαλύτερο θα είναι το πρόβλημα για τις συναρτήσεις κατακερματισμού – Γιατί;

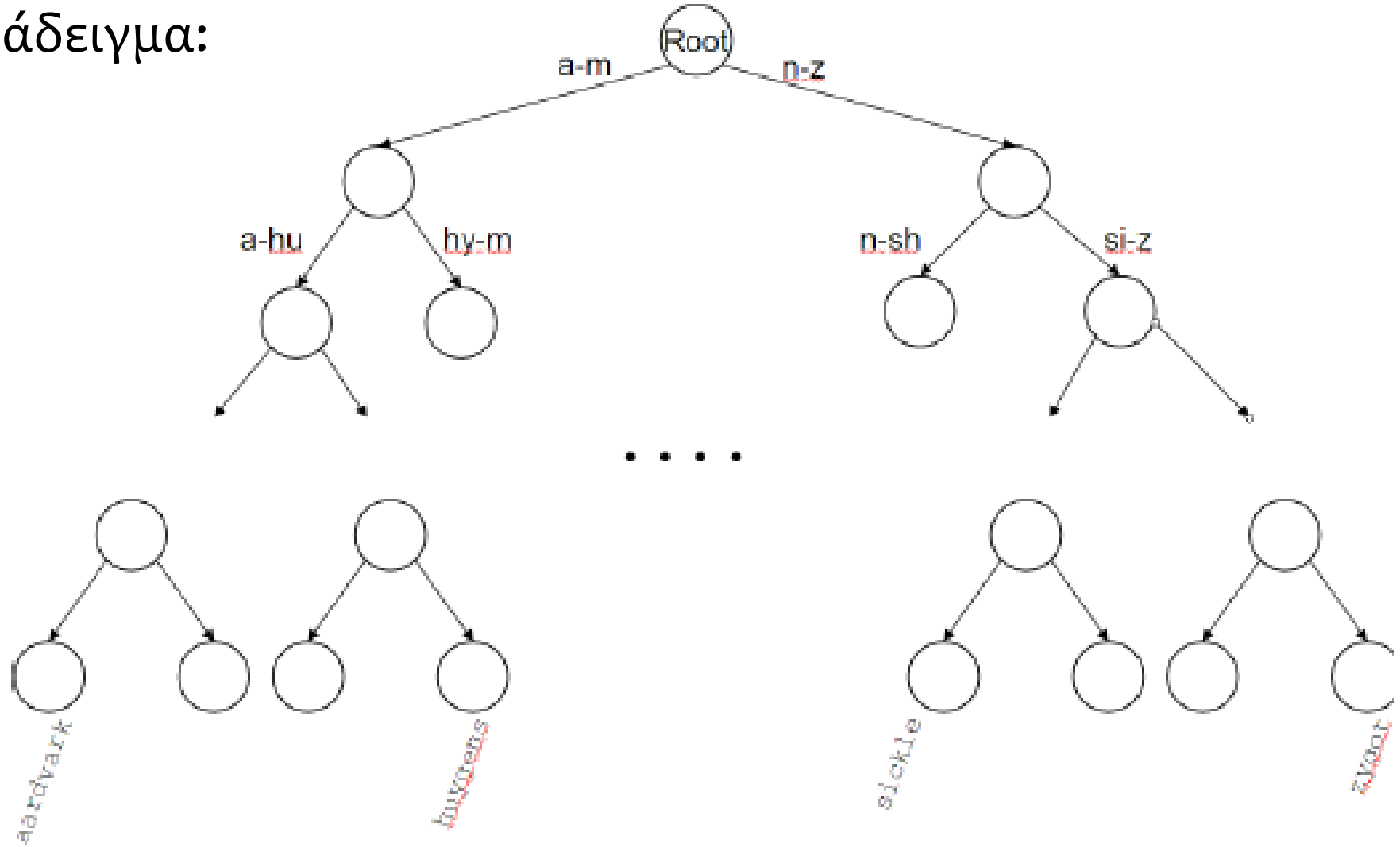
Δομές Αναζήτησης

- Τα δένδρα αναζήτησης ξεπερνούν πολλά από τα προηγούμενα προβλήματα
- Για παράδειγμα επιτρέπουν την απαρίθμηση όλων των όρων που έχουν το ίδιο πρόθεμα (π.χ., Δημοκρατ)
- Το πιο γνωστό είδος δένδρων είναι τα δυαδικά δένδρα αναζήτησης

- Η αναζήτηση ξεκινά από τη ρίζα
- Κάθε εσωτερικός κόμβος ουσιαστικά απεικονίζει ένα δυαδικό έλεγχο
- Το πρόβλημα εδώ έχει να κάνει με τις διαγραφές κόμβων
- Σε αυτές τις περιπτώσεις θα πρέπει το δένδρο να αναδιοργανωθεί
- Τα B-trees αποτελούν καλύτερη δομή

Δομές Αναζήτησης

- Παράδειγμα:



Ερωτήματα με Χρήση Wildcards

- Στα ερωτήματα μπορεί να χρησιμοποιήσουμε χαρακτήρες μπαλαντέρ για:
  - Τη διαχείριση της αβεβαιότητας (ο χρήστης δεν είναι σίγουρος για την πλήρη έκφραση)
  - Τη διαχείριση της έλλειψης γνώσης για τη συγγραφή κάποιων όρων
  - Την αναζήτηση εγγράφων που περιέχουν διάφορες εκφάνσεις των ό΄ρων
- Παράδειγμα:
  - Δημοκρατ* : trailing wildcard query
  - *μοκρατια : leading wildcard queries

Ερωτήματα με Χρήση Wildcards

- Τα δένδρα αναζήτησης είναι τα πιο κατάλληλα για τη διαχείριση των trailing wildcard queries
- Κατεβαίνουμε προς τα φύλλα του δένδρου
- Ακολουθούμε ένα ένα τα σύμβολα του κάθε όρου
- Φτάνουμε σε κάποιο κόμβο από όπου μπορούμε να έχουμε W όρους που ξεκινούν από το πρόθεμα που είναι πριν το wildcard
- Στο τέλος εκτελούμε W αναζητήσεις στο ευρετήριο για να πάρουμε όλα τα έγγραφα που περιλαμβάνουν τους W όρους

Ερωτήματα με Χρήση Wildcards

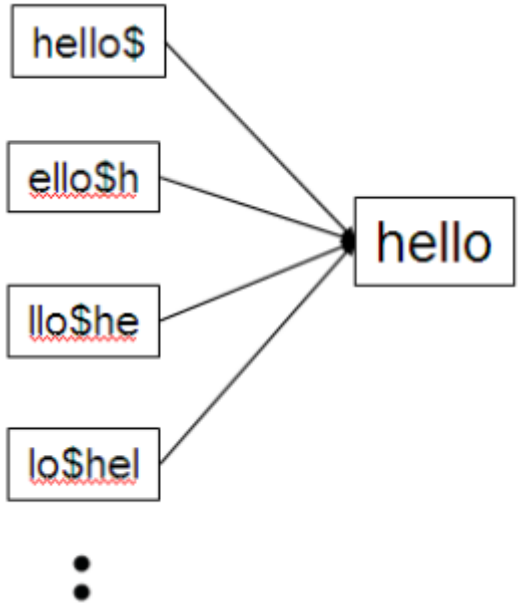
- Στα leading wildcard queries θεωρούμε ένα ανάστροφο B-δένδρο
- Κάθε μονοπάτι από τη ρίζα μέχρι τα φύλλα αντιστοιχεί σε ένα όρο του λεξικού που γράφεται ανάποδα
- Παράδειγμα:
  - Όρος: δήμος
  - Αναπαράσταση: ρίζα-ς-ο-μ-ή-δ
- Κάθε διαδρομή προς τα κάτω στο ανάστροφο B-δένδρο θα απαριθμήσει όλους τους όρους R που υπάρχουν στο λεξικό και θα έχουν κοινό πρόθεμα

Ερωτήματα με Χρήση Wildcards

- Αν υιοθετήσουμε ‘κανονικά’ B-δένδρα σε συνδυασμό με ανάστορφα B-δένδρα τότε μπορούμε να χειριστούμε καταστάσεις όπου τα wildcards βρίσκονται οπουδήποτε στην έκφραση
- Παράδειγμα:
  - Δη*τια
- Παίρνουμε ένα B-δένδρο για το πρόθεμα και ένα ανάστροφο για το επίθεμα
- Στη συνέχεια παίρνουμε την τομή των δύο συνόλων W & R
- Έπειτα από το ευρετήριο παίρνουμε όλα τα έγγραφα που περιέχουν τους όρους της τομής των συνόλων

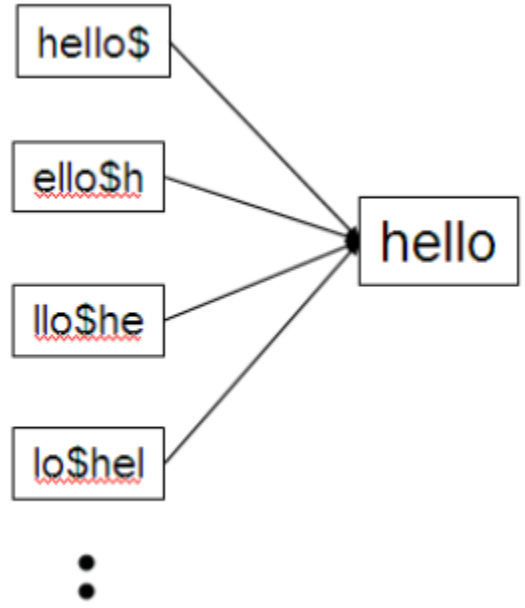
Permuterm Index

- Έχουμε το σύμβολο $ που υποδηλώνει το τέλος ενός όρου
- Ο όρος hello μπορεί να απεικονιστεί ως hello$
- Κατασκευάζουμε ένα permuterm index όπου βάζουμε διάφορες περιστροφές του κάθε όρου
- Όλα δείχνουν στον ίδιο όρο στο λεξικό



Permuterm Index

- Αναφερόμαστε σε όλους αυτούς τους όρους ως το permuterm λεξικό
- Η χρήση του γίνεται ως εξής:
  - Έστω το ερώτημα m*n
  - Θέλουμε να περιστρέψουμε το ερώτημα με τέτοιο τρόπο ώστε το * να έρθει στο τέλος της συμβολοσειράς
  - Οπότε γίνεται n$m*
  - Έπειτα ψάχνουμε το n$m* στο permuterm λεξικό και βρίσκουμε τον όρο (π.χ. θα βγάλει man, moron)
  - Όταν βρούμε τους όρους, τότε τους αναζητούμε στο αρχικό ευρετήριο για να πάρουμε τα έγγραφα



Permuterm Index

- Πιο σύνθετη περίπτωση αποτελεί το ερώτημα fi*mo*er
- Σε αυτή την περίπτωση λειτουργούμε ως ακολούθως:
  - Περιστρέφουμε το ερώτημα και παράγουμε το er$fi*
  - Αναζητούμε στο permuterm λεξικό το ερώτημα
  - Παίρνουμε όλους τους όρους
  - Εφαρμόζουμε ένα φίλτρο το οποίο ψάχνει αν το mo βρίσκεται στους όρους που ανακτήσαμε (προφανώς στη μέση)
  - Οι όροι που απομένουν τους αναζητούμε στο αρχικό ευρετήριο

Permuterm Index

- Μειονέκτημα:
  - Όσο περισσότερα permutations εφαρμόσουμε τόσο μεγαλύτερο θα είναι το permuterm λεξικό
  - Προσπαθούμε να παράξουμε όλα τα permutations των όρων

k-Gram Indexes

- Ένα k-gram είναι μια ακολουθία k χαρακτήρων
- Σε ένα ευρετήριο k-gram περιλαμβάνονται όλα τα k-grams που μπορεί να συναντήσουμε στο λεξικό
- Κάθε postings περιλαμβάνει ένα δείκτη από ένα k-gram προς όλους τους όρους που το περιέχουν

k-Gram Indexes

- Έστω ότι αναζητούμε το re*ve
- Εκτελούμε ένα Boolean ερώτημα $re AND ve$
- Θα κάνουμε αναζήτηση στο 3-gram ευρετήριο και θα πάρουμε όλους τους όρους του λεξικού που τα περιέχουν π.χ. remove, retrieve, relive
- Αναζητούμε κάθε ένα από τους προηγούμενους όρους στο αρχικό ευρετήριο για να πάρουμε τα έγγραφα

Mispellings

- Στοχεύουμε στη διόρθωση λαθών στα ερωτήματα

- Δύο είναι οι λύσεις:
  - Edit distance
  - k-gram overlap

Mispellings

- Για την πλειοψηφία των αλγορίθμων δύο είναι οι βασικές αρχές:
  - Μέσα στις πολλές εναλλακτικές διαλέγουμε την πιο κοντινή
  - Απαιτεί να έχουμε μια άποψη για το τι σημαίνει το να είναι κοντινές δύο εκφράσεις
  - Δύο ερωτήματα που είναι σωστά και συνδέονται είτε απόλυτα είτε σε μεγάλο βαθμό τότε επιλέγουμε το πιο κοινό
  - Παράδειγμα: grunt & grant είναι τα πιο κοντινά για το grnt
  - Επιλέγουμε αυτό που απαντάται τις περισσότερες φορές
  - Πολλές μηχανές επιλέγουν το ερώτημα που έχουν θέσει περισσότερες φορές άλλοι χρήστες

Mispellings

- Στην τεχνική της isolated-term διόρθωσης επιχειρούμε τη διόρθωση ενός όρου του ερωτήματος κάθε φορά
- Ο κάθε όρος ελέγχεται ξεχωριστά ακόμα και αν υπάρχουν περισσότεροι

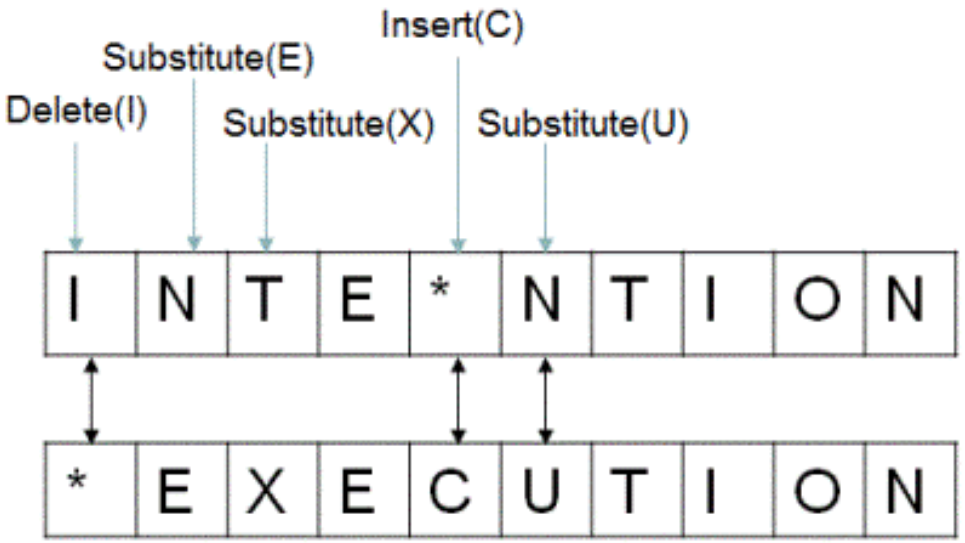
- Η τεχνική δεν μπορεί να εντοπίσει λάθη της ακόλουθης μορφής:
  - Flew form heathrow
- Ο λόγος είναι ότι ο κάθε όρος ξεχωριστά είναι σωστά διατυπωμένος

Mispellings

- Edit distance
  - Η edit distance ανάμεσα σε δύο αλφαριθμητικά s1, s2 είναι ο μικρότερος αριθμός επεξεργασιών για να μετατρέψουμε το ένα αλφαριθμητικό στο άλλο
    - Εισαγωγή χαρακτήρα
    - Διαγραφή χαρακτήρα
    - Αντικατάσταση χαρακτήρα
  - Ονομάζεται αλλιώς και Levenshtein distance
  - Παράδειγμα:
    - Cat – Dog: distance = 3
  - Μπορεί να γενικευτεί και να συμπεριλάβει βάρη για κάθε ενέργεια
  - Τα βάρη μπορεί να εξαρτώνται από τις πιθανότητες αντικατάστασης ενός γράμματος από ένα άλλο (π.χ. Phonetic similarity)

Mispellings

- Edit distance



Mispellings

- Edit distance
  - Χρησιμοποιούμε δυναμικό προγραμματισμό
  - Χρησιμοποιούμε ένα πίνακα |s1|X|s2|
  - Κάθε κελί (i,j) αποθηκεύει την απόσταση των αλφαριθμητικών μέχρι τον i χαρακτήρα του s1 και τον j χαρακτήρα του s2
  - Κάθε κελί περιέχει: κάτω δεξιά το ελάχιστο από τις άλλες τρεις θέσεις, και οι άλλες τρεις θέσεις έχουν τα m[i-1], m[j-1], m[i-1,j-1]+0 ή 1

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Mispellings

- Edit distance

		f		a		s		t	
	0	1	1	2	2	3	3	4	4
c	1	*1*	2	2	3	3	4	4	5
	1	2	*1*	2	2	3	3	4	4
a	2	2	2	*1*	3	3	4	4	5
	2	3	2	3	*1*	2	2	3	3
t	3	3	3	3	2	2	3	2	4
	3	4	3	4	2	3	2	3	2
s	4	4	4	4	3	2	3	3	3
	4	5	4	5	3	4	2	3	3

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Mispellings

- Edit distance

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Mispellings

- Edit distance
  - Υπολογίστε την edit distance για τα paris & alice
  - Λύση: 4

	a	l	i	c	e
p	1	2	3	4	5
a	1	2	3	4	5
r	2	2	3	4	5
i	3	3	2	3	4
s	4	4	3	3	4

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Mispellings

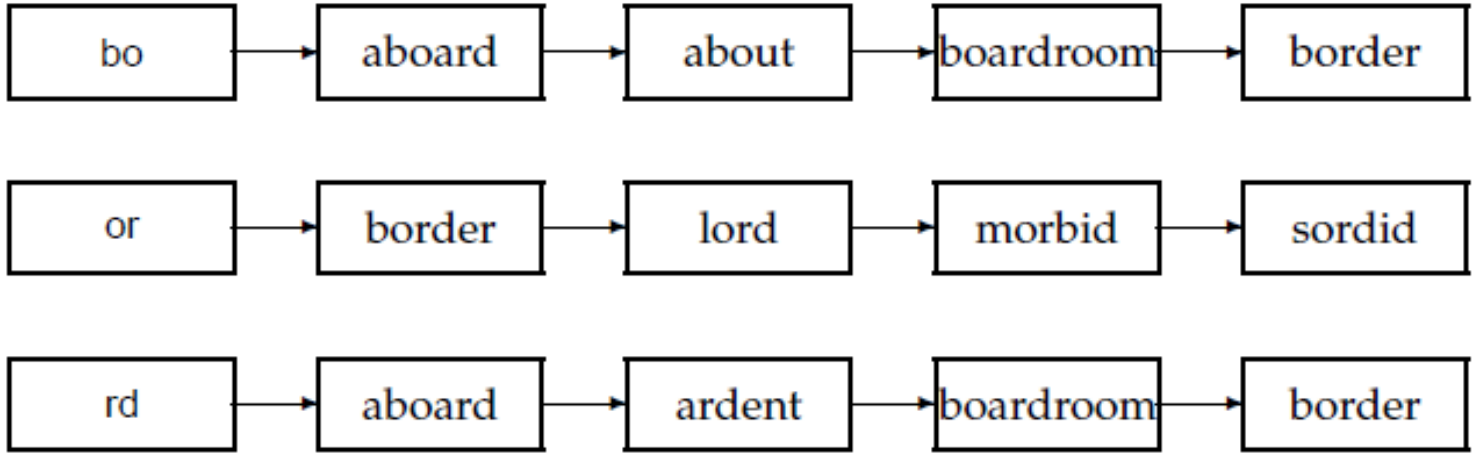
- Edit distance
  - Για να κάνουμε διορθώσεις με την τεχνική, εκτελούμε τα ακόλουθα:
    - Αναζητούμε τους όρους στο λεξικό που έχουμε τη μικρότερη απόσταση σε σχέση με το ερώτημα – πρόκειται για ένα exhaustive search πρόβλημα
    - Στη συνέχεια μπορούμε να:
      - Περιοριστούμε στους όρους που ξεκινούν με το ίδιο γράμμα όπως το ερώτημα
      - Εξετάσουμε όλες τις αναδιατάξεις του ερωτήματος και να υιοθετήσουμε το permuterm index

Mispellings

- K-gram overlap
  - Μπορούμε να χρησιμοποιήσουμε τα k-grams για να ανακτήσουμε όρους από το λεξικό που μοιράζονται πολλές κοινές ακολουθίες γραμμάτων με το ερώτημα
  - Στη συνέχεια μπορούμε να επιλέξουμε τους όρους με τη μικρότερη απόσταση από το ερώτημα
  - Προσπελαύνουμε τις postings έτσι ώστε να βρούμε τους όρους

Mispellings

- K-gram overlap
  - Παράδειγμα με 2-grams για το ερώτημα bord



  - Αν πάρουμε την τομή των λιστών (περιέχουν τουλάχιστον δύο 2-grams) έχουμε τα: border, abroad, boardroom
  - Το πρόβλημα είναι ότι προκύπτουν όροι που αποκλείεται να αποτελούν διορθωσεις του ερωτήματος π.χ. boardroom

Mispellings

- K-gram overlap
  - Χρειαζόμαστε πιο κατάλληλες τεχνικές πέραν της τομής των λιστών
  - Αντί για την τομή, μπορούμε να υιοθετήσουμε την τεχνική Jaccard
  - Για δύο λίστες A και B ο Jaccard coefficient υπολογίζεται ως εξής: $\frac{|A \cap B|}{|A \cup B|}$
  - Τα δύο σύνολα είναι το σύνολο των k-grams στο ερώτημα και το σύνολο των k-grams σε ένα όρο του λεξικού
  - Προσωρούμε από τον ένα όρο σε κάθε επόμενο κάθε φορά και υπολογίζουμε το συντελεστή Jaccard μεταξύ του ερωτήματος και του όρου
  - Αν ο συντελεστής υπερβαίνει ένα threshold τότε ο όρος λαμβάνει μέρος στα αποτελέσματα
  - Αν όχι, τότε προχωρούμε στον επόμενο όρο

Mispellings

- K-gram overlap
  - Το πρόβλημα σε αυτή την τεχνική έχει να κάνει με τον υπολογισμό των k-grams σε κάθε όρο του λεξικού
  - Πρακτικά πρόκειται για αργή διαδικασία αφού πρέπει για κάθε όρο να εξάγουμε τα k-grams on the fly
  - Για να διευκολύνουμε τη διαδικασία κάνουμε την εξής παρατήρηση: για να υπολογίσουμε το συντελεστή δεν χρειάζεται να υπολογίζουμε την ένωση των k-grams αφού το μήκος των συμβολοσειρών μας φτάνει
  - Παράδειγμα:
    - q=bord, t=boardroom
    - 2-grams: 2
    - JC=2/(8+3-2)=0.222
    - (boardroom -> μήκος 9 άρα 8 δι-γράμματα, bord -> μήκος 4 αρα 3 δι-γράμματα, 2 είναι τα κοινά δι-γράμματα)

Mispellings

- Τελικά η διόρθωση των ερωτημάτων γίνεται με:
  - Χρήση των k-index ευρετηρίων ώστε να ανακτηθούν οι πιο σχετικοί όροι
  - Εύρεση της edit distance με τους επιλεγμένους όρους
  - Επιλογή του όρου που έχει τη μικρότερη απόσταση από το ερώτημα

Mispellings

- Διόρθωση με βάση το πλαίσιο (context sensitive spelling correction)
  - Η τεχνική της isolated term διόρθωσης θα αποτύχει όταν θα γίνουν ορθογραφικά λάθη που οδηγούν όμως σε σωστές λέξεις π.χ. Flew form Heathrow
  - Ο πιο απλός τρόπος είναι να παραθέσουμε διορθώσεις για κάθε ένα από τα στοιχεία του ερωτήματος όπως έχουμε δει ακόμη και αν το ερώτημα δεν έχει λάθη
  - Στη συνέχεια δοκιμάζουμε αντικαταστάσεις στη φράση / ερώτημα
  - Παράδειγμα: fled form Hewthrow, flew fore Hewthrow, flew from Hewthrow
  - Η μηχανή αναζήτηση στη συνέχεια θα εκτελέσει το ερώτημα και θα πάρει αποτελέσματα ομοιότητας
  - Το πρόβλημα είναι ότι μπορεί να πάρουμε πολλές διορθώσεις για διαφόρους όρους του ερωτήματος
  - Μπορούμε όμως να διατηρήσουμε μόνο τους πιο συχνούς συνδυασμούς

Mispellings

- Διόρθωση με βάση το πλαίσιο (context sensitive spelling correction)
  - Οι συχνότητα των συνδυασμών μπορεί να εξαχθεί από τα logs των ερωτημάτων
  - Χρειάζεται προσοχή διότι τα ερωτήματα αυτά μπορεί να περιλαμβάνουν λάθη
  - Για παράδειγμα μπορεί να διατηρηθεί το flew from αντί των flew form & flew fore
  - Αφού βρούμε τους συχνούς συνδυασμούς για π.χ. δύο όρους, επεκτείνουμε και στους υπόλοιπους για να βρούμε αντίστοιχους συνδυασμούς

Phonetic Correction

- Η ιδέα είναι να δημιουργήσουμε για κάθε όρο ένα φωνητικό κατακερματισμό (phonetic hash)
- Όροι που ακούγονται όμοια θα κατακερματίζονται στην ίδια τιμή
- Οι αλγόριθμοι που υιοθετούνται είναι γνωστοί ως soundex αλγόριθμοι
- Ο αλγόριθμος – βάση έχει ως εξής:
  - Μετατροπή κάθε όρου ώστε να απεικονιστεί σε μια μορφή 4-χαρακτήρων – χτίζουμε ένα inverted index από αυτές τις μικρότερες μορφές (soundex index)
  - Κάνουμε το ίδιο και για το ερώτημα
  - Όταν ψάχνουμε ένα ταίριασμα για το ερώτημα, τότε αναζητούμε στο soundex index

Phonetic Correction

- Οι διαφορές με τους αλγορίθμους για phonetic correction εστιάζονται στη χρήση των 4-χαρακτήρων
- Μια κοινή μέθοδος είναι ο πρώτος χαρακτήρας να είναι γράμμα και τα υπόλοιπα 3 να είναι ψηφία

Phonetic Correction

- Μέθοδος:
  - Διατηρούμε το πρώτο γράμμα του όρου
  - Αλλάζουμε όλα τα γράμματα A, E, I, O, U, H, W, Υμε 0
  - Αλλάζουμε:
  - B, F, P, V σε 1
  - C, G, J, K, Q, S, X, Z σε 2
  - D,T σε 3
  - L σε 4
  - M, N σε 5
  - R σε 6
  - Διαγράφουμε επαναληπτικά ένα από κάθε ζεύγος συνεχόμενων ίδιων ψηφίων
  - Διαγράφουμε όλα τα 0 από το τελικό αποτέλεσμα (βάζουμε στο τέλος αν απαιτείται) – παίρνουμε τις 4 πρώτες θέσεις
  - Παράδειγμα: Hermann -> H065055 -> H655

Phonetic Correction

- A, E, I, O, U, H, W, Υμε 0
- B, F, P, V σε 1
- C, G, J, K, Q, S, X, Z σε 2
- D,T σε 3
- L σε 4
- M, N σε 5
- R σε 6

- Βρείτε τα indexes για τα Washington & Lee
- W252 & L000

Phonetic Correction

- Τα φωνήεντα δεν επηρεάζουν
- Σύμφωνα με παρόμοιο ήχο / προφορά μπαίνουν στις ίδιες κλάσεις