13장 — 검색어 자동완성 시스템

단어를 입력하다 보면 입력 중인 글자에 맞는 검색어가 자동으로 완성되어 표시되는 시스템

문제 이해, 설계 범위 확정

질문 (요구사항 확립)

- 사용자의 입력이 자동완성의 첫 부분만 가능 OR 중간 부분도 될 수 있음
- 자동완성 되어 표시되는 단어의 수
- 영어 OR 한국어 OR 다국어
- 대문자, 특수 문자 처리
- 사용자의 수, MAU, DAU

고려 해야할 사항

- 응답 속도
- 연관성
- 정렬
- 규모 확장성, 고가용성
- QPS

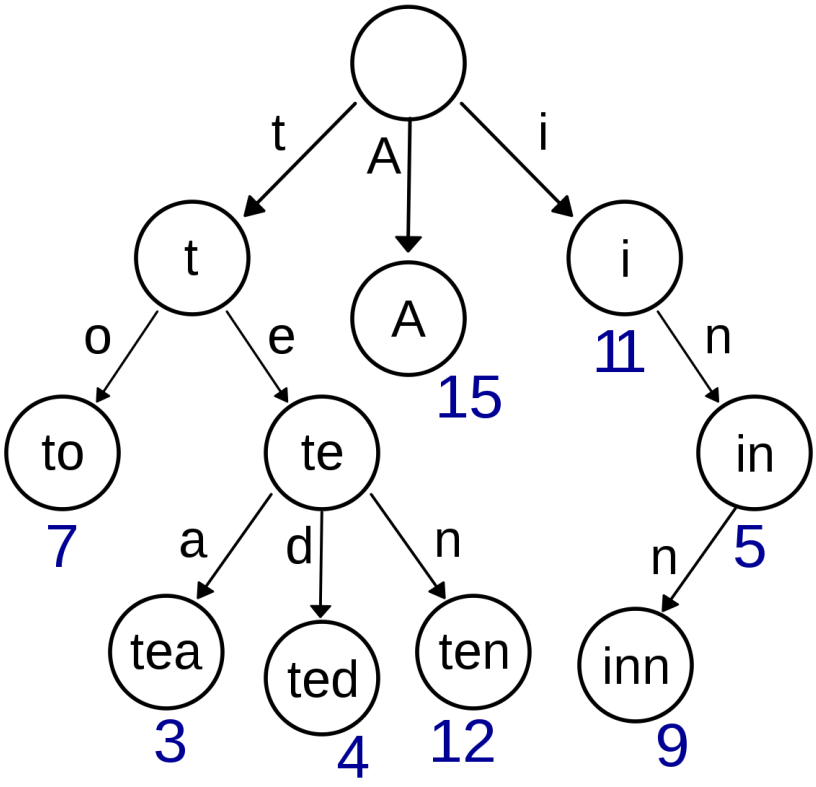
개략적 설계안

- 데이터 수집 서비스
  - 사용자의 입력을 수집하는 시스템
  - 질의문, 빈도수 저장
- 질의 서비스
  - 질의에 대해 인기 검색어를 정렬해 주는 시스템

상세 설계

트라이 자료구조

- 문자열을 저장하고 효율적으로 탐색하기 위한 `트리` 형태의 자료구조 (prefix tree)



- 탐색 시간, 검색어 k개를 찾는다. O(p) + O(c) + O($clogc$)
  1. 해당 접두어를 표현하는 노드를 찾는다. → O(p), p는 prefix의 길이
  2. 해동 노드로부터 시작하는 하위 트리를 탐색, 모든 유효 노드를 찾는다. → O(c), c는 트라이 안에 있는 노드 개수
  3. 유효 노드들 중 정렬하여 검색어를 k개 찾는다. → O($clogc$), c는 트라이 안에 있는 노드 개수
- 최악의 경우 트라이 전체를 다 검색해야 하는 일이 생길 수 있다.
  - ?? 접두어의 최대 길이를 제한 ??
  - 노드에 인기 검색어를 캐시

접두어 최대 길이 제한

매우 긴 검색어를 입력하는 일은 별로 없기 때문에 접두어 최대 길이를 제한한다.

- 최대길이를 50으로 제한했을 경우: O(p) → O(50) = O(1)

노드에 인기 검색어 캐시

- 노드 마다 인기 질의어를 캐시
  - 응답속도 향상
  - 저장공간 희생
- O($clogc$) → O(1)

데이터 수집 서비스

특징

- 실시간으로 트라이를 갱신하면 캐싱이 거의 불가능 → 속도 매우 저하
- 트위터 같이 실시간 검색어를 제공하는 것이 아니라면, 트라이는 자주 갱신할 필요가 없다.

데이터 수집

1. 서비스 로그 저장 → 사용자들의 질의 로그를 저장한다.
2. 모든 로그를 저장하기에는 너무 데이터가 많을 수도 있다. → 데이터 샘플링 (N개의 질의 중 1개만 추출해 로그 저장)
3. 데이터가 너무 많을 경우 샤딩
   a. 알파벳 첫글짜로 샤딩
      i. 단어를 시작하는 알파벳으로 샤딩시 단어가 고르게 나눠지지 않는다.
4. 로그 취합 → 외부 작업 서버(배치 서버) 에서 로그를 취합한다. → 이후 트라이 갱신
   a. 쿼리와, 주간 검색 횟수 등

작업 서버

- 취합된 데이터를 트라이로 만든다.

    - 트라이를 만들고 DB에 저장
        - Document Store: MonogoDB 같이 문서로 저장
        - Key-Value: 질의어를 key로
    - 트라이 캐시
        - 주간 1회 트라이를 갱신한면, 이후 트라이를 캐싱한다.
- 트라이 갱신
    - 기존 트라이 대체
    - 각 노드를 갱신
        - 데이터가 많으면 매우 성능이 떨어짐

질의 서비스

- AJAX 요청 → 당연히 페이지 새로 고침하지 않고 api 콜 필요
- 브라우저 캐싱 가능
- API 서버에서 캐시에 데이터를 가져올 때 중간에 필터 계층을 두고 필터링 가능
    - 혐어 표현, 성적 표현, 커스텀 필터 등 검색 결과 제어 가능

그 외 고려해볼 만한 사항

- 유니 코드 고려
- 국가별 순위 지원 → CDN
- 최신 검색어 가중치