UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Ingegneria Elettrica e delle tecnologie dell'informazione

Secondo Homework del corso di

BIG DATA ENGINEERING

Professore:
Giancarlo Sperlì

Candidati:
Benfenati Domenico M63001165
Carandente Vincenzo M63001229

ANNO ACCADEMICO 2021/2022
Secondo Semestre

Indice

Capitolo 1

Introduzione

1.1 Business Understanding

Tale elaborato ha l'obiettivo di processare un insieme di dati relativi alle recensioni effettuate dagli utenti riguardo alcuni locali e ristoranti all'interno del dataset **Yelp**.
Lo scopo dell'elaborato è quello di elaborare i dati all'interno del dataset, sfruttando l'architettura di **Hadoop** tramite il suo top-level **Pig**.
L'obiettivo del processamento effettuato è quello di mettere in evidenza informazioni esplicite, sfruttando query e algoritmi sviluppati in linguaggio **Pig Latin**.

1.2 Data Understanding

Nella sezione corrente si vogliono descrivere con maggiore precisione i dati a disposizione, i quali possono essere divisi in differenti categorie: **Business**, **Review**, **Users**, **Checkin**, **Tips**. È bene sottolineare che, dovendo implementare un database documentale, i dati utilizzati per popolarlo provengono da file in formato *.json*.

1.2.1 Business

La prima categoria di dati consiste in 150346 istanze, e include tutte le informazioni relative agli esercizi commerciali che sono inclusi all'interno delle classifiche Yelp.
In particolare, il file considerato nell'elaborazione è *yelp_academic_dataset_business.json* e contiene informazioni riguardo i vari esecizi commerciali quali posizione, città, valutazione in numero di stelle, numero di recensioni ricevute, e altre informazioni riguardo gli "attributi" del locale.

1.2.2 Review

La seconda categoria di dati contiene circa 6990280 istanze, e include tutte le informazioni relative alle recensioni sulla piattaforma Yelp. In particolare il file è *yelp_academic_dataset_review.json* e contiene l'identificativo, l'id dell'utente e dell'esercizio commerciale, la data, il numero di stelle attribuite, il testo della recensione e dei interi che specificano il numero di voti utili, divertenti e hot che altri utenti hanno attributo a tale recensione.

1.2.3 User

La terza categoria di dati conta circa 1987897 documenti, e include tutte le informazioni relative agli utenti presenti all'intenro della piattaforma Yelp.
In particolare il file è *yelp_academic_dataset_user.json* e contiene l'identificativo dell'utente, il nome, la data d'iscrizione, il numero di recensioni da esso effettuate, una lista di amici ad esso collegati, e una serie di valori interi indicanti il conteggio di differenti tipologie di complimenti ricevuti dall'utente. Data la numerosità dei dati, si è scelto di non considerare tale categoria ai fini dell'analisi.

1.2.4 Checkin

La quarta categoria di dati aggrega circa 131930 istanze, e include tutte le informazioni relative ai check-in degli utenti all'interno degli esercizi commerciali.
In particolare il file è *yelp_academic_dataset_checkin.json* e contiene l'identificativo del locale e una lista di date corrispondenti ai checkin effettuati in quell'esercizio. Tale categoria non è risultata utile ai fini dell'analisi, e si è quindi scelto di non considerare tali dati.

1.2.5 Tip

La quinta categoria di dati conta 908915 elementi, e include tutte le informazioni relative ai suggerimenti inseriti dagli utenti all'interno della piattaforma Yelp.
In particolare il file considerato nell'elaborazione è *yelp_academic_dataset_tip.json* e contiene informazioni circa l'utente che ha inserito il suggerimento, il locale a cui è destinato, e la data di inserimento, nonchè il testo in chiaro inserito dall'utente. Tale categoria non è risultata utile ai fini dell'analisi, e si è quindi scelto di non considerare tali dati.

Capitolo 2

Data Processing

Nel capitolo corrente sono riportate le operazioni di processing effettuate sui dati al fine di estrarne informazioni utili.

2.1 Pig

Per il processing dei dati si è deciso di utilizzare **Apache Pig**: esso è una piattaforma di alto livello per creare programmi MapReduce da usare con Apache Hadoop. Pig Latin è il linguaggio sfruttato da Pig, il quale astrae la programmazione dall'idioma **Java MapReduce** in una notazione che rende la programmazione MapReduce di alto livello in maniera simile all'SQL dei sistemi RDBMS. Tramite la shell di Pig, GRUNT, è possibile definire dei comandi SQL-like sequenziali, per la definizione di variabili e per il processamento dei dati.

2.2 Lettura dei File

Attraverso la shell di Pig, GRUNT, si vanno a leggere i file in ingresso, Business e Review, tramite le seguenti operazioni:

```
1 Review = LOAD '/home/domybenf/Scrivania/review.json' USING JsonLoader('review_id:
      CHARARRAY,user_id:CHARARRAY,business_id:CHARARRAY,stars:FLOAT,useful:INT,funny:INT,
      cool:INT,text:CHARARRAY,date:DATETIME');
2 Business = LOAD '/home/domybenf/Scaricati/yelp_academic_dataset_business.json' USING
      JsonLoader('business_id:CHARARRAY,name:CHARARRAY,address:CHARARRAY,city:CHARARRAY,
      state:CHARARRAY,postal_code:CHARARRAY,latitude:FLOAT,longitude:FLOAT,stars:FLOAT,
      review_count:INT,is_open:INT,attributes:MAP[],categories:CHARARRAY,hours:MAP[]');
```

Per verificare che il caricamento sia avvenuto correttamente, è possibile generare la descrizione dei tipi importati, tramite i seguenti comandi:

```
1 DESCRIBE Review;
2 DESCRIBE Business;
```

L'output ottenuto è il seguente:

```
Review: {review_id:CHARARRAY, user_id:CHARARRAY, business_id:CHARARRAY,
↪  stars:FLOAT, useful:INT, funny:INT, cool:INT, text:CHARARRAY, date:DATETIME}
Business: {business_id:CHARARRAY, name:CHARARRAY, address:CHARARRAY,
↪  city:CHARARRAY, state:CHARARRAY, postal_code:CHARARRAY, latitude:FLOAT,
↪  longitude:FLOAT, stars:FLOAT, review_count:INT, is_open:INT,
↪  attributes:MAP[], categories:CHARARRAY, hours:MAP[]}
```

Si nota come la descrizione del file combaci con il caricamento effettuato, e si conclude quindi che il caricamento sia andato a buon fine.

2.3 Operazioni effettuate

2.3.1 Conteggio recensioni migliori per stato

Si è voluto verificare quante fossero le recensioni a cinque stelle per i business di uno stato di esempio come la California (indicata con CA all'intero della collection Business).
Si procede quindi ad effettuare un primo filtraggio delle recensioni e dei business, rispettivamente per numero di stelle pari a 5 e per stato pari a CA, tramite i seguenti comandi:

```
1 Business_filtered = FILTER Business BY (state=='CA');
2 Review_filtered = FILTER Review BY (stars==5.0);
```

Successivamente si va ad effettuare l'operazione di JOIN tra le collection filtrate grazie all'attributo `business_id`, tramite il seguente comando:

```
1 Review_join_business = JOIN Business_filtered BY business_id, Review_filtered BY
      business_id;
```

Infine è necessario effettuare un raggruppamento di tutte le collection ed effettuarne successivamente un conteggio, tramite i comandi seguenti:

```
1 group_rev_join_bus = GROUP Review_join_business ALL;
2 group_count = FOREACH group_rev_join_bus GENERATE COUNT($1);
```

Si noti che l'espressione `$1` indica il secondo parametro dell'output generato tramite l'operazione di GROUP, all'interno del quale sono inserite tutte le recensioni ottenute dalla JOIN.
Tale esecuzione porta al seguente output:

```
(190206)
```

Si ottiene che il numero di recensioni a cinque stelle per tutti gli esercizi commerciali dello stato della California sono **190206**.

2.3.2 Generazione del Vocabolario delle recensioni

Si vuole costruire il vocabolario dei termini utilizzati all'interno delle recensioni.
Per rendere l'esecuzione dei task Map e Reduce computazionalmente meno onerosa, e rendere l'output più snello, si è scelto di limitare i documenti in ingresso tramite il comando:

```
Review_lim = LIMIT Review 30;
```

A valle di tale operazione si procede a separare i termini che sono presenti all'interno delle review, raggrupparli ed effettuarne un conteggio, tramite le operazioni seguenti:

```
RAW = FOREACH Review_lim GENERATE review_id,text;
testo = FOREACH RAW GENERATE text;
parole = FOREACH testo GENERATE FLATTEN(TOKENIZE(REPLACE(LOWER(TRIM($0)),  '[\\p{Punct
    },\\p{Cntrl}]',''))) as parola;
gruppo_parole = GROUP parole BY parola;
conteggio = FOREACH gruppo_parole GENERATE group, COUNT(parole);
```

Si passa infine ad ordinare tali termini per frequenza di occorrenza all'interno dei documenti, e si effettua un salvataggio del vocabolario ottenuto in un file apposito denominato `voc.txt`, tramite i seguenti comandi:

```
ordine = ORDER conteggio BY $1 DESC;
vocabolario = FOREACH ordine GENERATE $0;
STORE vocabolario INTO 'voc.txt'
```

2.3.3 Conteggio parole positive e negative

Dato il vocabolario, si vogliono analizzare i termini all'interno dello stesso, verificando il numero di parole positive e negative contenute all'interno del subset di recensioni utilizzato.
Per effettuare tale analisi vengono presi in considerazione i termini della collection **WordNet**. WordNet è un **database semantico-lessicale** per la lingua inglese, che si propone di organizzare, definire e descrivere i concetti espressi dai vocaboli.
Di tale database ne è stata eseguita una pre-elaborazione, fino all'identificazione di una collection che ne racchiude i termini corredandoli di un coefficiente di positività e di negatività, variabile a seconda del **synset**, cioè del legame semantico di tale termine con altri che ne rappresentano un sinonimo.
L'elaborazione genera due file contenenti le parole in cui predomina il coefficiente di positività e di negatività, rispettivamente `senti_words_positive.txt` e `sent_words_negative.txt`. Si procede quindi ad effettuare il caricamento di tali file testuali tramite le seguenti operazioni:

```
positivi = LOAD '/home/domybenf/Scaricati/senti_words_positive.txt' USING TextLoader AS
    pos_words:CHARARRAY;
negativi = LOAD '/home/domybenf/Scaricati/senti_words_negative.txt' USING TextLoader AS
    neg_words:CHARARRAY;
```

Successivamente si procede ad identificare all'interno del vocabolario quali e quante sono le parole che compaiono all'interno dei vocabolari positivi e negativi, tramite il seguente set di comandi:

```
voc_join_pos = JOIN vocabolario BY $0, positivi BY $0;
voc_join_neg = JOIN vocabolario BY $0, negativi BY $0;
pos_terms = GROUP voc_join_pos ALL;
neg_terms = GROUP voc_join_neg ALL;
cont_pos = FOREACH pos_terms GENERATE COUNT($1);
cont_neg = FOREACH neg_terms GENERATE COUNT($1);
DUMP cont_pos;
DUMP cont_neg;
```

L'esecuzione di tali istruzioni porta all'identificazione del seguente output:

```
(42)
(39)
```

I documenti analizzati contengono quindi **42** termini identificati come positivi e **39** identificati come negativi.
Per poter rielaborare tali termini in altro modo, si è scelto di effettuare un operazione di salvataggio di tali termini, tramite i seguenti comandi:

```
top30_pos_words = FOREACH voc_join_pos GENERATE $0;
top30_neg_words = FOREACH voc_join_neg GENERATE $0;
STORE top30_pos_words INTO 'top30_pos_words';
STORE top30_neg_words INTO 'top30_neg_words';
```

2.3.4 Occorrenze dei termini positivi

A valle dell'individuazione dei termini positivi e negativi, si è deciso di contare le occorrenze di tali termini all'interno delle prime trenta recensioni.
Come prima operazione è necessario effettuare un JOIN tra le parole contenute all'interno delle recensioni e i termini positivi individuati, tramite il seguente comando:

```
pos_terms = JOIN top30_pos_words BY $0, parole BY $0;
```

Successivamente è necessario raggruppare per parole tale output, sfruttando le seguenti operazioni:

```
pos_terms_group = GROUP pos_terms BY $0;
pos_terms_count = FOREACH pos_terms_group GENERATE $0, COUNT($1);
por_terms_occurrency = ORDER pos_terms_count BY $1 DESC;
top10_pos_terms_occurrency = LIMIT pos_terms_occurrency 10;
```

Si ottiene quindi il seguente elenco di parole in output:

```
(up,17)
(good,16)
(like,14)
(time,12)
(well,8)
(better,7)
(nice,6)
(think,6)
(see,5)
(worth,5)
```

2.3.5 Termini positivi per ogni recensione

Si sceglie ora di verificare l'occorrenza dei termini positivi trovati in precedenza all'interno di ogni recensione. Per le successive analisi si è scelto di limitare le recensioni alle prime dieci, e raggrupparne i termini per recensione, tramite i seguenti comandi:

```
rev_lim = LIMIT Review 10;
words_per_rev = FOREACH rev_lim GENERATE review_id, FLATTEN(TOKENIZE(REPLACE(LOWER(TRIM(
    text)),'[\\p{Punct},\\p{Cntrl}]','')));
rev_words = GROUP words_per_rev BY review_id;
rev_words = FOREACH rev_words GENERATE $0, $1.$1;
```

Da tale esecuzione si genera l'output composto dall'identificativo della review e l'elenco delle parole in essa contenute; per completezza si riporta un esempio di tale output:

```
(-1vQd1dCGvwP8oqluUDPNw,{(my),(review),(is),(not),(based),(on),(their),
↪  (adoption),(services),(however),(as),(a),(store),(where),
↪  (i),(shop),(for),(my),(pup),(the),(staff),(is),
↪  (helpful),(and),(friendly),(they),(also),(are),
↪  (very),(knowledgeable),(and),(carry),(a),(great),(line),(of),(products),
↪  (its),(a),(good),(store)})
```

Successivamente si procede ad effettuare il caricamento del file delle parole positive trovate in precedenza tramite il comando:

```
pos_rev_words = LOAD '/home/domybenf/hadoop/pig/bin/top30_pos_words/part-r-00000' USING
    TextLoader AS pos_rev_word:CHARARRAY;
```

Si vuole poi controllare, per ogni recensione, quali siano le parole buone in essa contenute, ed effettuarne il conteggio, tramite i comandi:

```
join_words = JOIN words_per_rev BY $1, pos_rev_words BY $0;
gruppo_parole_rev = GROUP join_words BY review_id;
Parole_buone = FOREACH gruppo_parole_rev GENERATE group, COUNT($1);
```

A valle di tale operazione, si ottiene il seguente output:

```
(--ez98beazUDWG2sf07J_A,15)
(-1vQd1dCGvwP8oqluUDPNw,1)
(-2ms6jEf6a3onw72OiA3Mw,2)
(-4tkPyUEtrS8N5xSEVPKkA,30)
(-5R9KicW7i4O4L8xwX1inQ,2)
(-6tmyRgMVlwANGCn2YBXIA,4)
(-9VWjlUMxBMHXgGaKsXo1A,1)
(-B6aQC5xJOXzSg0RAWyg3A,23)
(-BQnbwIbNLRHFeQUHfU7fA,4)
```

Si noti come l'output prodotto contiene unicamente 9 reviews; questo è dovuto probabilmente al fatto che tra le prime 10 reviews considerate, una di esse non contiene alcun termine positivo, e di conseguenza l'operazione di JOIN non ha considerato tale recensione.

2.3.6 Numero termini positivi VS Valutazione

Si vuole ora verificare che un numero di termini positivi elevato corrisponda effettivamente ad una valutazione elevata della recensione, espressa in termini di numero di stelle.
L'operazione da fare è quindi effettuare l'unione tra le recensioni e quelle contenenti termini positivi, tramite il loro identificativo, e effettuare una proiezione dei soli campi utili, sfruttando i seguenti comandi:

```
parole_buone_reviews = JOIN Parole_buone BY $0, rev_lim BY review_id;
Num_parol_buone_VS_stelle = FOREACH parole_buone_reviews GENERATE review_id, $1, stars;
Num_parol_buone_VS_stelle = ORDER Num_parol_buone_VS_stelle BY $1 DESC;
```

Si ottiene quindi l'output seguente:

```
(-4tkPyUEtrS8N5xSEVPKkA,30,4.0)
(-B6aQC5xJOXzSg0RAWyg3A,23,5.0)
(--ez98beazUDWG2sf07J_A,15,3.0)
(-6tmyRgMVlwANGCn2YBXIA,4,1.0)
(-BQnbwIbNLRHFeQUHfU7fA,4,5.0)
(-2ms6jEf6a3onw72OiA3Mw,2,4.0)
(-5R9KicW7i4O4L8xwX1inQ,2,4.0)
```

```
(-1vQd1dCGvwP8oqluUDPNw,1,5.0)
(-9VWjlUMxBMHXgGaKsXo1A,1,5.0)
```

2.3.7 Term Frequency - Inverse Document Frequency

Per terminare l'analisi riguardo i termini all'interno delle recensioni, si è deciso di verificare in quanti documenti compaiono i vari termini.

Tale analisi è fatta per un calcolo empirico della frequenza inversa nei documenti: infatti, per valutare quanto un termine, detto anche **token** sia rappresentativo per un documento, è necessario che quel termine sia ripetuto molte volte all'interno del documento, ma non sia un termine comune a tutta la collezione documentale.

La metrica che infatti è necessario calcolare per verificare l'importanza di un termine è detta **term-frequency-inverse-document-frequency**, derivante dal fatto che voglio massimizzare la frequenza del termine nel documento, e minimizzarla tra tutti i documenti. Considerando quindi una review Yelp come un documento, si vuole calcolare tale metrica per i termini delle prime dieci review.

Il primo passo da compiere è quindi contare le occorrenze dei termini in ogni documento, e non solo tra tutti i documenti, tramite i comandi:

```
1 doc_tokens = GROUP words_per_rev BY ($0,$1);
2 doc_token_counts = FOREACH doc_tokens GENERATE FLATTEN(group), COUNT(words_per_rev) AS
      num_doc_tok_usages;
```

Successivamente si va a calcolare la prima metrica, **Term Frequency**, la frequenza del termine all'interno dei documenti, normalizzando tale metrica per la lunghezza del documento, tramite i seguenti comandi:

```
1 doc_usage_bag = GROUP doc_token_counts BY review_id;
2 doc_usage_bag_fg = FOREACH doc_usage_bag GENERATE group AS doc_id, FLATTEN(
      doc_token_counts.(token, num_doc_tok_usages)) AS (token, num_doc_tok_usages), SUM(
      doc_token_counts.num_doc_tok_usages) AS doc_size;
3 term_freqs = FOREACH doc_usage_bag_fg GENERATE doc_id, token AS token, ((double)
      num_doc_tok_usages/(double)doc_size) AS term_freq;
```

Si riporta quindi un estratto dell'output ottenuto, contenente identificativo della recensione, token, e frequenza del token:

```
(-J0Db6QIyloSh8bPmSaW4w,i,0.0784313725490196)
(-J0Db6QIyloSh8bPmSaW4w,in,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,my,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,of,0.0196078431372549)
```

```
(-J0Db6QIyloSh8bPmSaW4w,and,0.058823529411764705)
(-J0Db6QIyloSh8bPmSaW4w,did,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,for,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,the,0.13725490196078433)
(-J0Db6QIyloSh8bPmSaW4w,was,0.0392156862745098)
(-J0Db6QIyloSh8bPmSaW4w,fast,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,food,0.0392156862745098)
(-J0Db6QIyloSh8bPmSaW4w,from,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,okay,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,pats,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,size,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,this,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,will,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,again,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,boxed,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,combo,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,enjoy,0.0196078431372549)
(-J0Db6QIyloSh8bPmSaW4w,fries,0.058823529411764705)
```

Successivamente è necessario contare in quanti documenti compare ogni token, per poter poi calcolare la document frequency inversa, tramite i comandi:

```
1 term_usage_bag = GROUP term_freqs BY token;
2 token_usages = FOREACH term_usage_bag GENERATE FLATTEN(term_freqs) AS (doc_id,token,
      term_freq), COUNT(term_freqs) AS num_docs_with_token;
```

Tale istruzione produce un append all'output precedente del numero di documenti contenenti quel token; per completezza se ne riporta un estratto:

```
(-B6aQC5xJOXzSg0RAWyg3A,already,0.0011312217194570137,1)
(-5R9KicW7i4O4L8xwX1inQ,alright,0.0078125,1)
(-B6aQC5xJOXzSg0RAWyg3A,another,0.0011312217194570137,1)
(-B6aQC5xJOXzSg0RAWyg3A,artisan,0.0022624434389140274,1)
(-5R9KicW7i4O4L8xwX1inQ,awesome,0.0078125,1)
(-B6aQC5xJOXzSg0RAWyg3A,because,0.003393665158371041,2)
(-4tkPyUEtrS8N5xSEVPKkA,because,0.0018018018018018018,2)
```

Infine, si procede quindi al calcolo della metrica TF-IDF completa. Indicando con D_{tot} il numero totale di documenti e con D_j in numero di documenti che contengono il termine j, si effettua il

calcolo della inverse document frequency tramite la seguente formula:

$$IDF(j) = \log \frac{D_{tot}}{D_j} \tag{2.1}$$

Successivamente, la metrica TF-IDF è calcolata modulando la frequenza del termine per la sua inverse document frequency, secondo la formula:

$$TF - IDF_j = TF(j) * IDF(j) \tag{2.2}$$

Di seguito si riportano le istruzioni in Pig delle operazioni precedenti:

```
tfidf_all = FOREACH token_usages {
        idf = LOG((double)10/(double)num_docs_with_token);
        tf_idf = (double)term_freq*idf;
        GENERATE doc_id, token, tf_idf AS TF_IDF;
        };
```

Si è quindi deciso di ordinare i termini in base a tale metrica ed effettuarne un salvataggio, tramite i comandi:

```
order_tfidf = ORDER tfidf_all BY TF_IDF DESC;
STORE order_tfidf INTO 'TF_IDF';
```

Filtrando infine tali elementi, per una determinata recensione, si può verificare come ci siano parole che risultano più caratterizzanti, e parole che invece presentano una metrica pari o prossima allo zero. Tali termini sono identificabili come **stopwords** e non aggiungono alcun valore a tale caratterizzazione. I termini che presentano un valore di metrica alta rappresentano i token che sono più specifici per quella recensione. Di seguito i comandi descritti e il relativo output:

```
filtered_tfidf = FILTER order_tfidf BY (doc_id == '-J0Db6QIyloSh8bPmSaW4w');
```

```
(-J0Db6QIyloSh8bPmSaW4w,fries,0.13544618194082622)
(-J0Db6QIyloSh8bPmSaW4w,burger,0.09029745462721749)
(-J0Db6QIyloSh8bPmSaW4w,ordered,0.04721461977748769)
(-J0Db6QIyloSh8bPmSaW4w,pats,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,delaware,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,weekend,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,burgers,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,neatly,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,amount,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,okay,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,delivery,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,enjoy,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,boxed,0.04514872731360874)
```

```
(-J0Db6QIyloSh8bPmSaW4w,standard,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,fast,0.04514872731360874)
(-J0Db6QIyloSh8bPmSaW4w,combo,0.03155760612615883)
(-J0Db6QIyloSh8bPmSaW4w,while,0.03155760612615883)
(-J0Db6QIyloSh8bPmSaW4w,did,0.03155760612615883)
(-J0Db6QIyloSh8bPmSaW4w,size,0.03155760612615883)
(-J0Db6QIyloSh8bPmSaW4w,overall,0.03155760612615883)
(-J0Db6QIyloSh8bPmSaW4w,food,0.027182242374899815)
(-J0Db6QIyloSh8bPmSaW4w,will,0.023607309888743846)
(-J0Db6QIyloSh8bPmSaW4w,order,0.023607309888743846)
(-J0Db6QIyloSh8bPmSaW4w,again,0.017966484938708924)
(-J0Db6QIyloSh8bPmSaW4w,from,0.010016188701293935)
(-J0Db6QIyloSh8bPmSaW4w,this,0.010016188701293935)
(-J0Db6QIyloSh8bPmSaW4w,was,0.00875072750251803)
(-J0Db6QIyloSh8bPmSaW4w,i,0.008263569855515792)
(-J0Db6QIyloSh8bPmSaW4w,my,0.004375363751259015)
(-J0Db6QIyloSh8bPmSaW4w,in,0.004375363751259015)
(-J0Db6QIyloSh8bPmSaW4w,for,0.002065892463878948)
(-J0Db6QIyloSh8bPmSaW4w,of,0.002065892463878948)
(-J0Db6QIyloSh8bPmSaW4w,the,0.0)
(-J0Db6QIyloSh8bPmSaW4w,and,0.0)
```