BIG DATA :

- dane, które nie mieszczą się w pamięci
- średnie dane, ale potrzeba stałego nie mieści się w pamięci

$$|P(\Omega)| = 2^{|\Omega|}$$

Ω

"Hello - world" dla Big Data

Word-count problem

X =  tekst

1) podziel X na słowa
2) wyznacz częstotliwości wszystkich słów
3) wyznacz z tego N najczęściej występujących słów ($N \approx 100$)

Zadanie: (4) wygenerować z (3) "word cloud"

Detale:

- input: ulubiona książka napisana w formacie txt [UTF-8]
- eliminacja śmieci: usuwamy znaki przystankowe/specjalne ',', '.', ';', '&', ':'
- stop-words: usuwamy to

Y = 

Python: $Z = Y.split(" ").$ ← lista słów

Z = ["ala", "ma", "kota", "i", "ala", "ma", "psa"]

W = [("ala", 2), ("ma", 2), ...]

Python: GroupBy ← lista posortowana

(TF-IDF) $\mathfrak{D} = \{D_1, D_2, \ldots, D_n\}$ ← kolekcja $\neq \emptyset$ dokumentów

T = zbiór słów wyst. w $\mathfrak{D}$.

Term frequency: $tf(w, d) = \frac{n_{w,d}}{\sum_{\alpha \in T} n_{\alpha,d}}$

($w \in T$, $d \in \mathfrak{D}$)

$n_{w,\alpha}$ = liczba wystąpień.

$d = [w_1, w_2, \ldots, w_n]$

$n_{w,d} = d.\text{count}(w)$:

$tf(w,d) =$ frequency word w in doc d.

$$idf(t, \mathcal{D}) = \log_2 \left(\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|} \right)$$

inverse doc. freq

$$1 = \frac{|\mathcal{D}|}{|\mathcal{D}|} \leq \frac{|\mathcal{D}|}{|\{d : t \in d\}|} \leq \frac{|\mathcal{D}|}{1} = |\mathcal{D}|$$

$\neq 0$

$$0 \leq idf(t, \mathcal{D}) \leq \log_2 |\mathcal{D}|$$

$$tfidf(t, d, \mathcal{D}) = tf(t, d) \cdot idf(t, \mathcal{D})$$

MAP - REDUCE

mapper:

on get x:

k - klucz wyjsc. z x

y - wartosc wyjsc z x

output (k, y)

programmer

$h: \Sigma \longrightarrow \{0, \ldots, n-1\}$

$(k, y) \rightsquigarrow R_{h(k)}$

PO ZAKOŃCZENIU PRACY PRZEZ MAPPERY

1) $(k_1, y_1), (k_2, y_2), (k_3, y_3), \ldots$

$\Downarrow$

$[(k^1, [y^1_1, \ldots, y^1_{k_1}]), (k^2, [y^2_1, \ldots, y^2_{k_2}]), \ldots]$

$[(a, x), (b, y), (a, z), (a, e), (b, w), (c, v)]$

$[(a, [x, z, e]), (b, [y, w]), (c, [v])]$

2) dla każdej pary (k, L) wykonywane jest obliczenie i wynikiem jest końcowy wynik. } poprawność

PRZYKŁAD: Mnożenie macierzy przez wektor

$$\underbrace{\begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ \vdots & & & \\ m_{w1} & m_{w2} & \cdots & m_{wn} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}}$$

zał.: $\vec{x}$ mieści się w pamięci Mappeduce

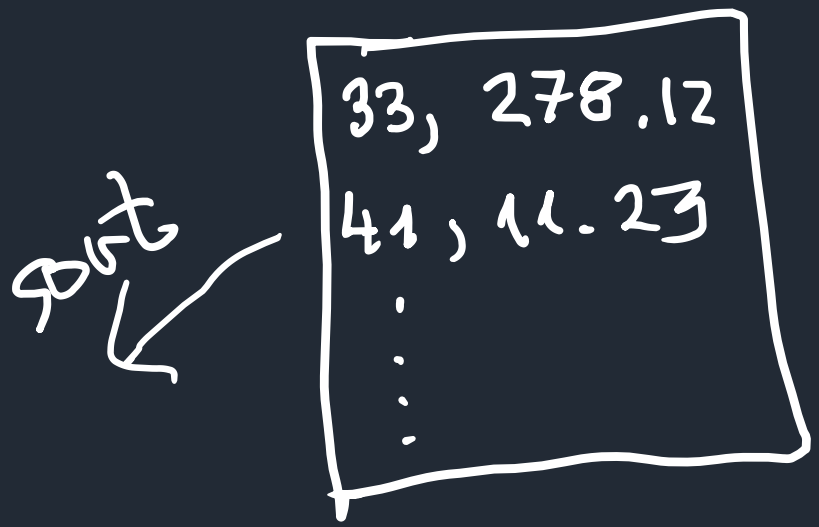
$$M \cdot \vec{x} = \vec{y}$$

INPUT: (i, j, m_{ij})

$$y_i = \sum_j m_{ij} \cdot x_j$$

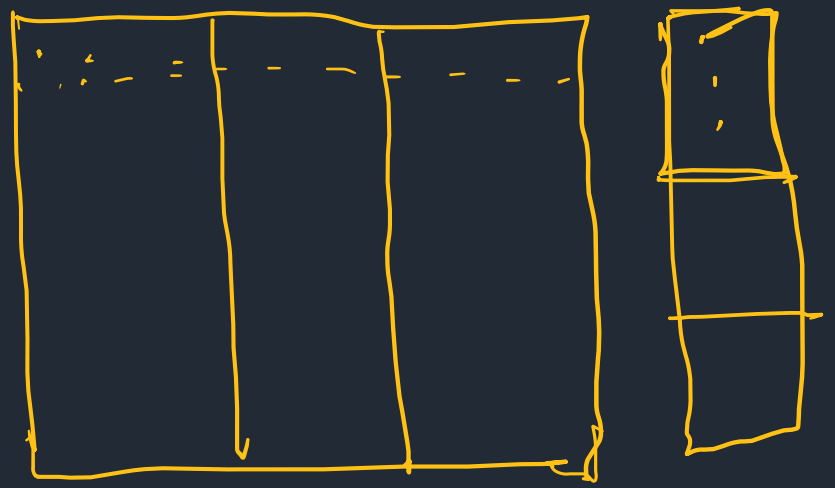
```
mapper (i, j, m) {
    emit ( i , m · x_j );
}

reduce (i, L) {
    emit ( i , sum(L) )
}
```



$(i, [m_{i1} \cdot x_1, m_{i2} \cdot x_2 \cdots])$

Tabourie: rat. ze $\overline{X}$ nie wiesce się
w pewnej mopperoles.
ale $\frac{1}{3}|\overline{X}|$ wesci się



jak to zrobić?

word - count.

```
mapper(w) {
    emit(w, 1);
}

reduce(W, L)
    emit(w, length(L));
}
```

reducer

(a, 1), (b, 1) (a, 1), (b, 1) (c, 1)

[(a, [1, 1, 1]), (b, [1, 1]) (c, [1]

wynik : plik

a, 3
c, 1
b, 2