

- CS341 info session is on Thu 3/18 7pm in Gates104
- Final exam logistics

# Optimizing Submodular Functions

CS246: Mining Massive Datasets  
Jure Leskovec, Stanford University  
<http://cs246.stanford.edu>



# Announcement: Final Exam Logistics

# Final: At Stanford

- **Alternate final: Fri 3/14 7:00-10:00pm** in Cubberley Auditorium
- **Final: Mon 3/17 12:15-3:15pm**  
**NVidia** (Lastname starting with A-M)  
**GatesB01** (Lastname starting with N-Z)
  - See <http://campus-map.stanford.edu>
  - Practice finals + Gradiance quizzes are on Piazza
  - Open book, open computer, no internet
- **SCPD students can take the exam at Stanford!**

# Final: SCPD Students

- **Exam protocol for SCPD students:**
  - On **Friday 3/14** your exam proctor will receive the PDF of the final exam from SCPD
  - **If you take the exam at Stanford:**
    - Ask the exam monitor to delete the SCP email
  - **If you don't take the exam at Stanford:**
    - Arrange a **3h** slot with your exam monitor
    - You can take the exam **anytime** but return it in time
    - **Email exam PDF to [cs246.mmds@gmail.com](mailto:cs246.mmds@gmail.com) by Tuesday 3/15 11:59pm Pacific time**

# Announcement: CS341: Project in Mining Massive Datasets

- **Data mining research project on real data**
  - Groups of 3 students
  - **We provide interesting data, computing resources (Amazon EC2) and mentoring**
  - **You provide project ideas**
  - Class meets once a week + individual group mentoring

**Information session:**  
**Tuesday 3/18 7:00pm in Gates 104**  
(there will be pizza!)

# CS341: Schedule

- **Tue 3/18: Info session**
  - We will introduce datasets, problems, ideas
- **Students form groups and project proposals**
- **Mon 3/24: Project proposals are due**
- **We evaluate the proposals**
- **Mon 3/31: Admission results**
  - 10 to 15 groups/projects will be admitted
- **Mon 5/5, Wed 5/7: Midterm presentations**
- **Thu 6/10: Presentations, poster session**

**More info: <http://cs341.stanford.edu>**

# Optimizing Submodular Functions

CS246: Mining Massive Datasets  
Jure Leskovec, Stanford University  
<http://cs246.stanford.edu>





# Recommendations: Diversity

- Redundancy leads to a bad user experience

**Obama Calls for Broad Action on Guns**

**Obama unveils 23 executive actions,  
calls for assault weapons ban**

**Obama seeks assault weapons ban,  
background checks on all gun sales**

- Uncertainty around information need => don't put all eggs in one basket
- How do we optimize for diversity directly?

[illegible]

## Hagel expects fight

10

[illegible]

## New gun proposals

11

# Encode Diversity as Coverage

- **Idea:** Encode diversity as coverage problem
- **Example:** Word cloud of news for a single day
  - Want to select articles so that most words are “covered”

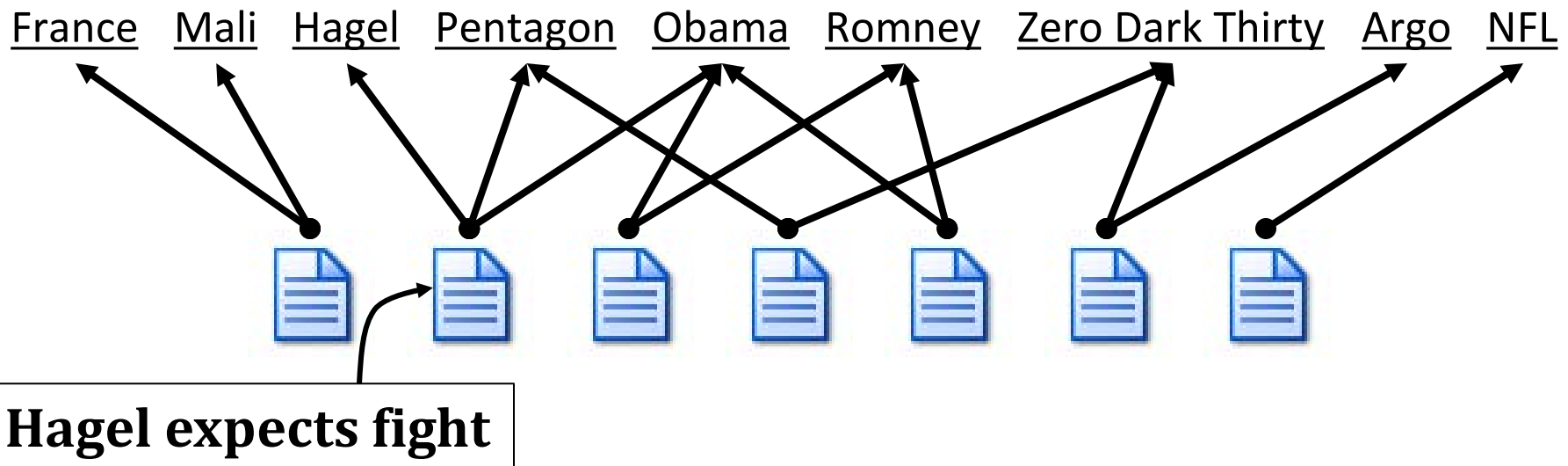




# Diversity as Coverage

# What is being covered?

- **Q: What is being covered?**
- **A: Concepts** (In our case: Named entities)



- **Q: Who is doing the covering?**
- **A: Documents**

# Simple Abstract Model

- Suppose we are given a set of documents  $V$ 
  - Each document  $d$  covers a set  $X_d$  of words/topics/named entities  $W$

- For each set of documents  $A$  we define

$$F(A) = \left| \bigcup_{d \in A} X_d \right|$$

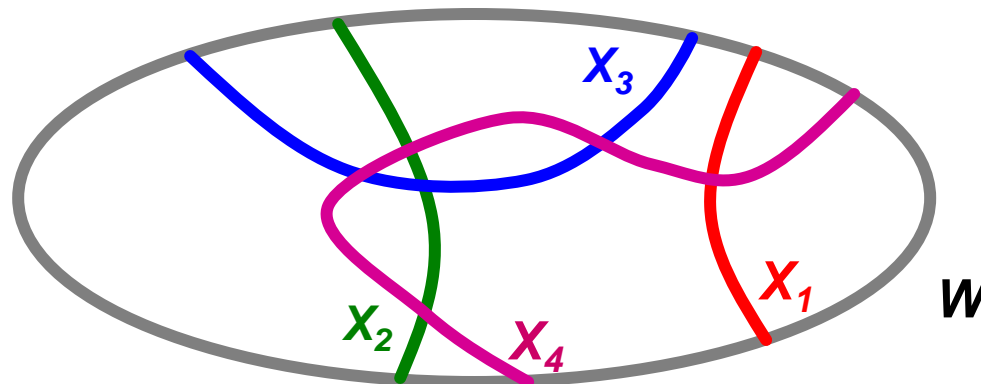
- Goal: We want to

$$\max_{|A| \leq k} F(A)$$

- Note:  $F(A)$  is a set function:  $F(A): \text{Sets} \rightarrow \mathbb{N}$

# Maximum Coverage Problem

- Given universe of elements  $W = \{w_1, \dots, w_n\}$  and sets  $X_1, \dots, X_m \subseteq W$



- Goal: Find  $k$  sets  $X_i$  that cover the most of  $W$ 
  - More precisely: Find  $k$  sets  $X_i$  whose size of the union is the largest
  - Bad news: A known NP-complete problem



# Simple Greedy Heuristic

## Simple Heuristic: Greedy Algorithm:

- Start with  $A_0 = \{ \}$
- For  $i = 1 \dots k$ 
  - Take set  $d$  that  $\max F(A_{i-1} \cup \{d\})$
  - Let  $A_i = A_{i-1} \cup \{d\}$

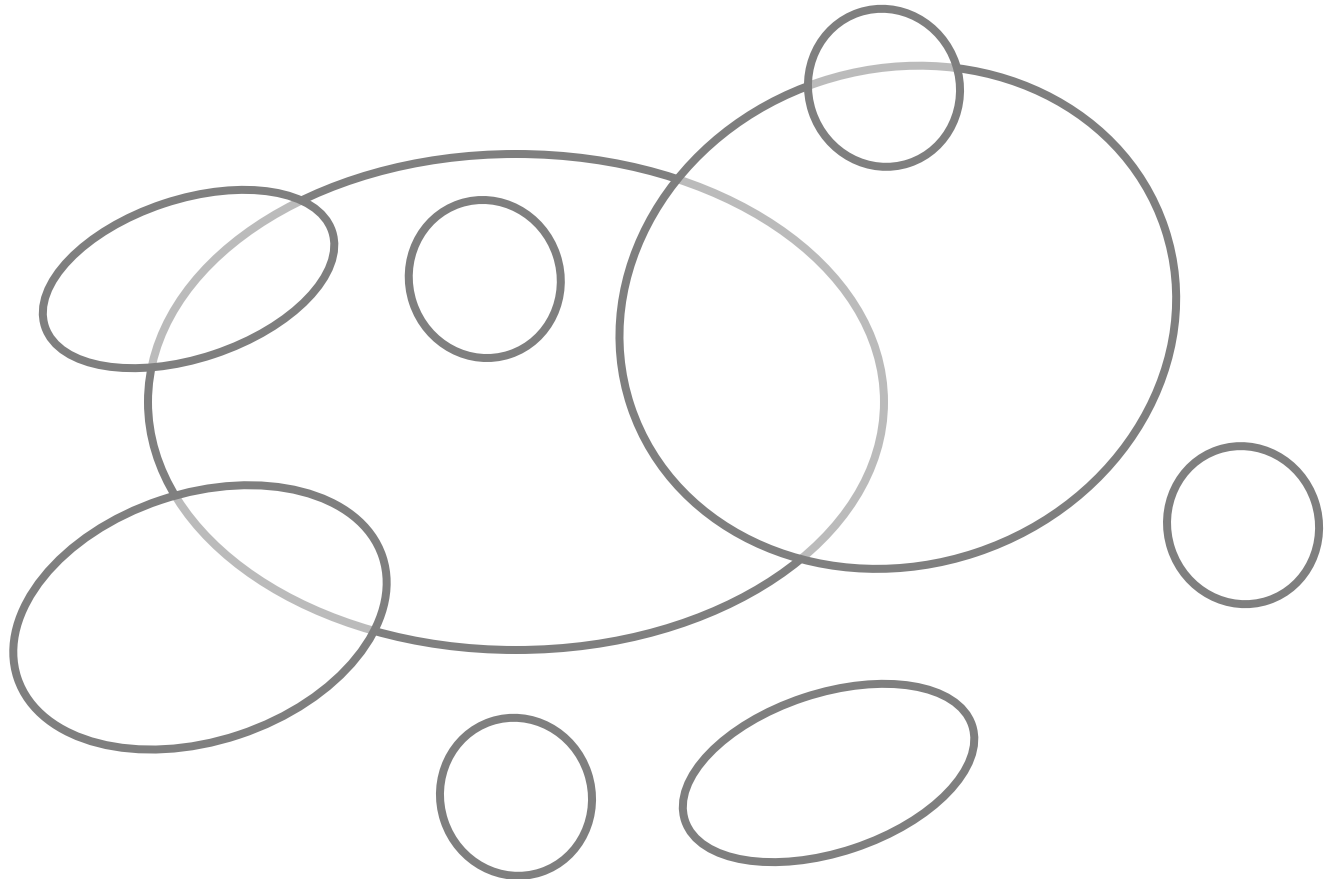
$$F(A) = \left| \bigcup_{d \in A} X_d \right|$$

## ■ Example:

- Eval.  $F(\{d_1\}), \dots, F(\{d_m\})$ , pick best (say  $d_1$ )
- Eval.  $F(\{d_1\} \cup \{d_2\}), \dots, F(\{d_1\} \cup \{d_m\})$ , pick best (say  $d_1$ )
- Eval.  $F(\{d_1, d_2\} \cup \{d_3\}), \dots, F(\{d_1, d_2\} \cup \{d_m\})$ , pick best
- And so on...

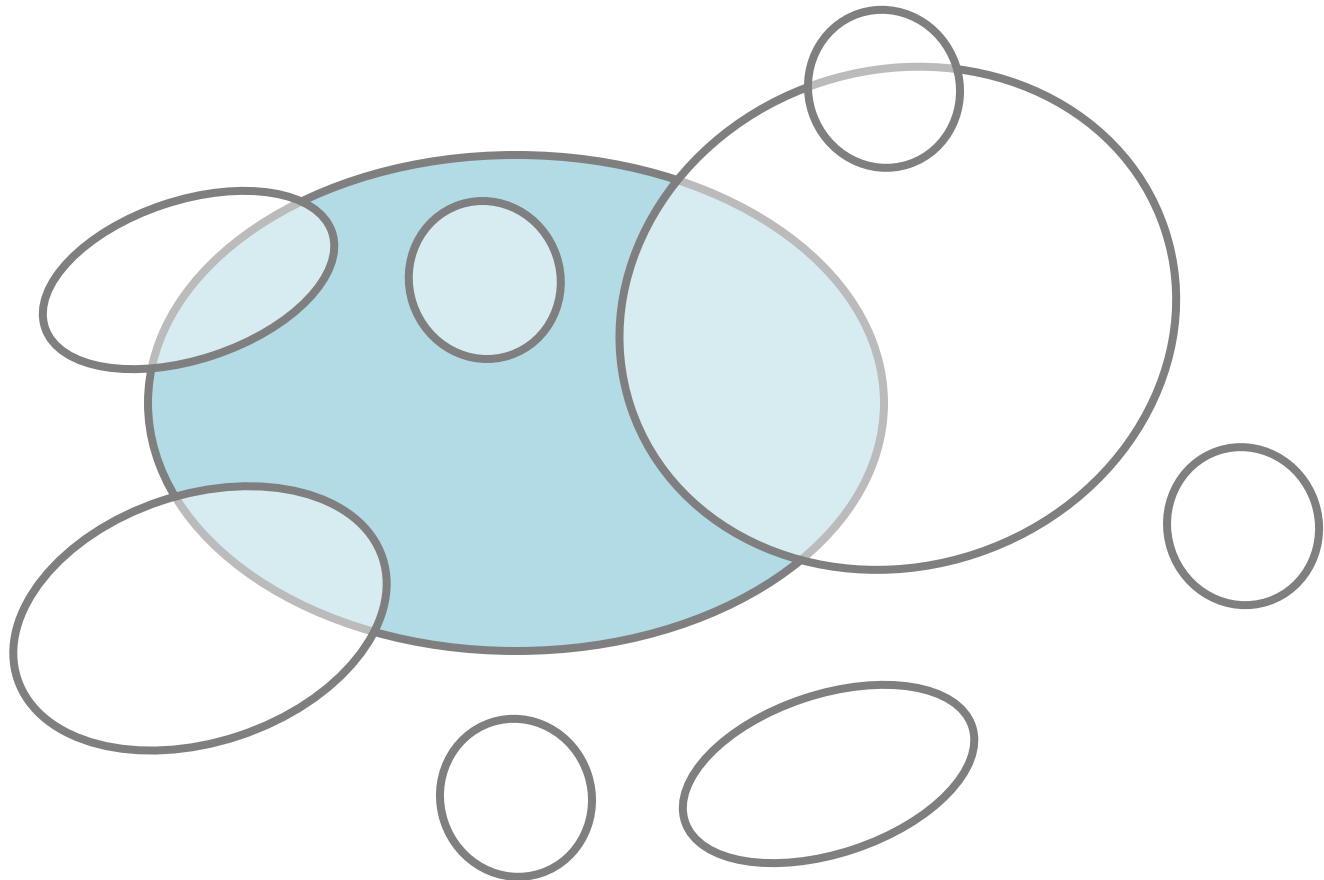
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



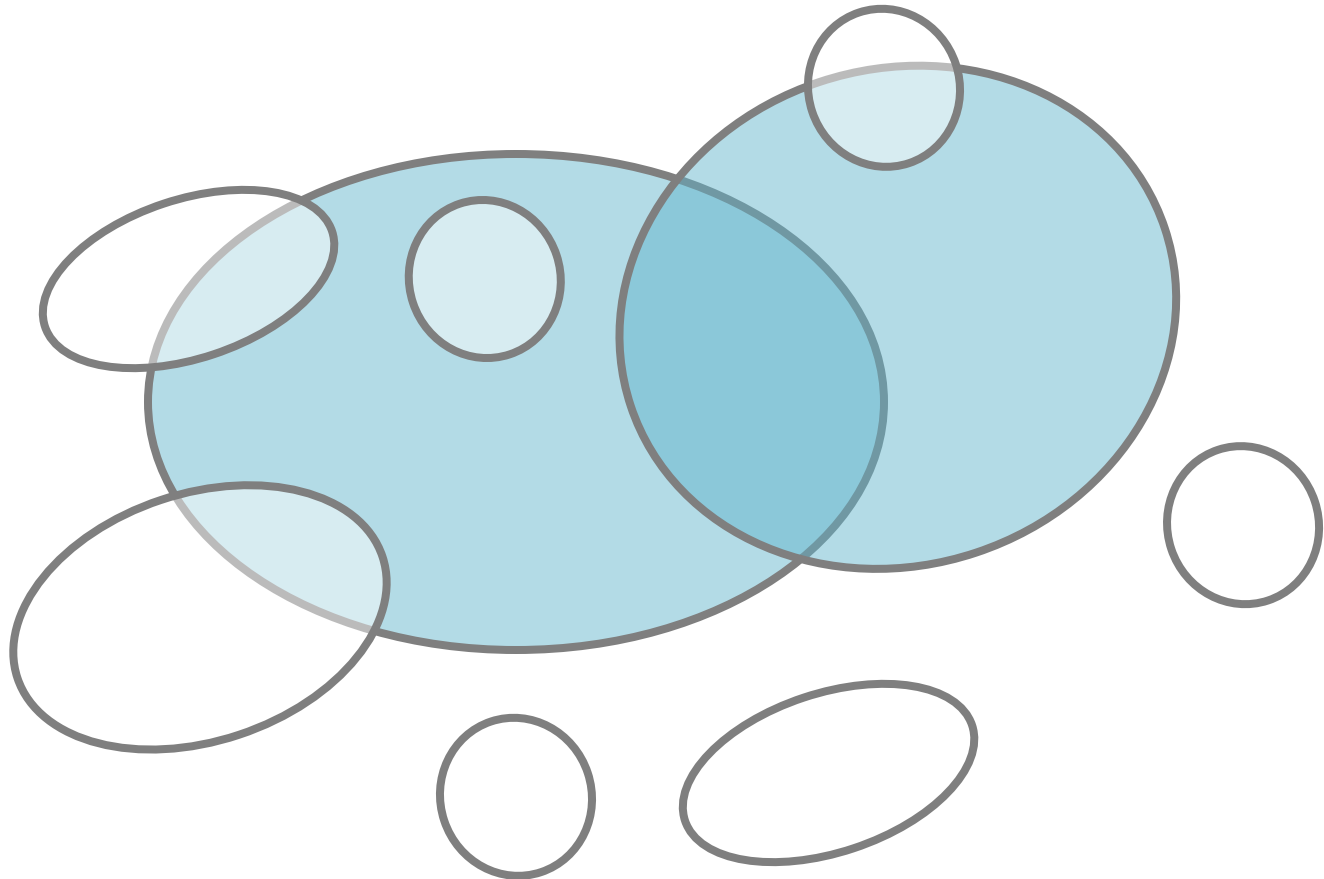
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



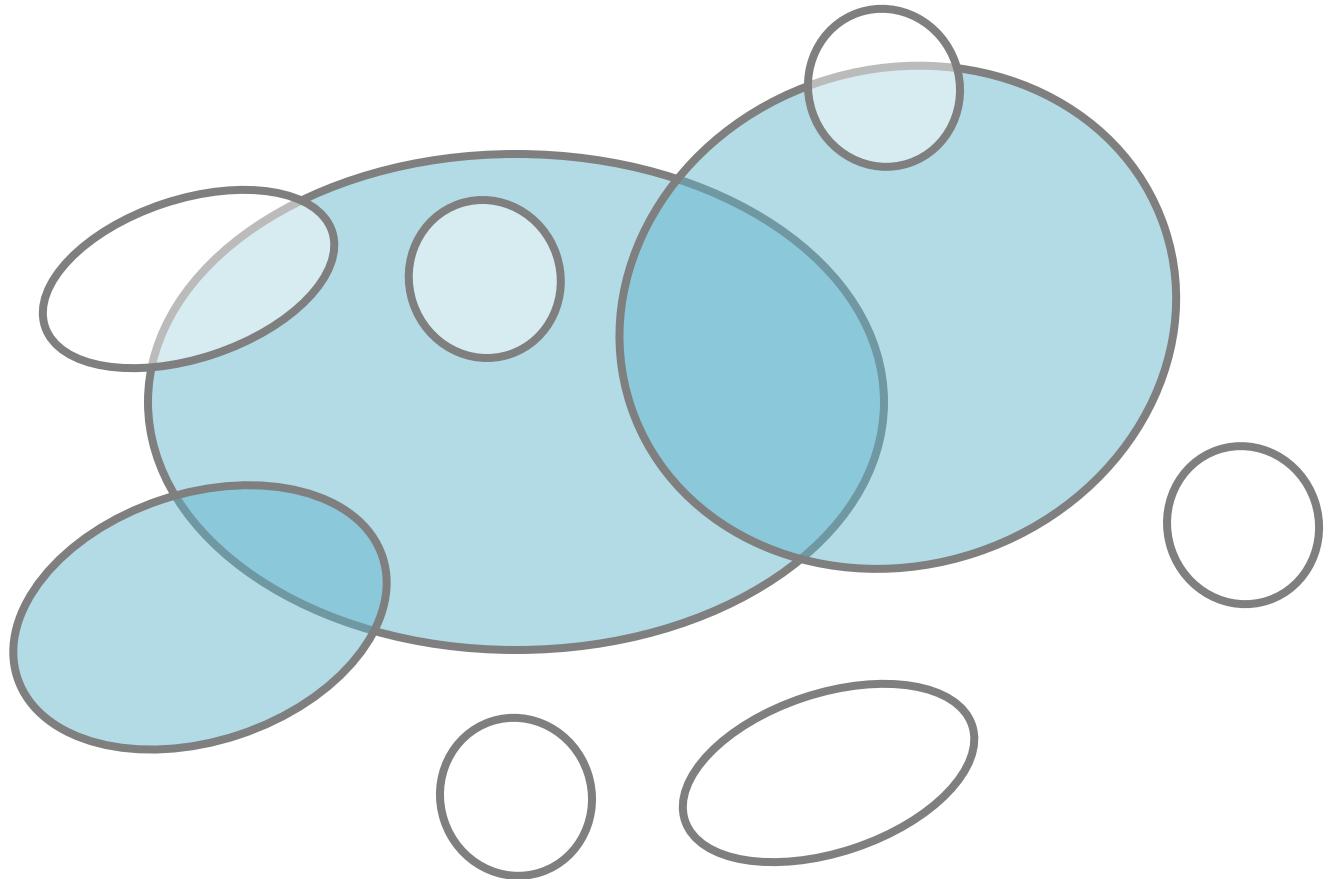
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



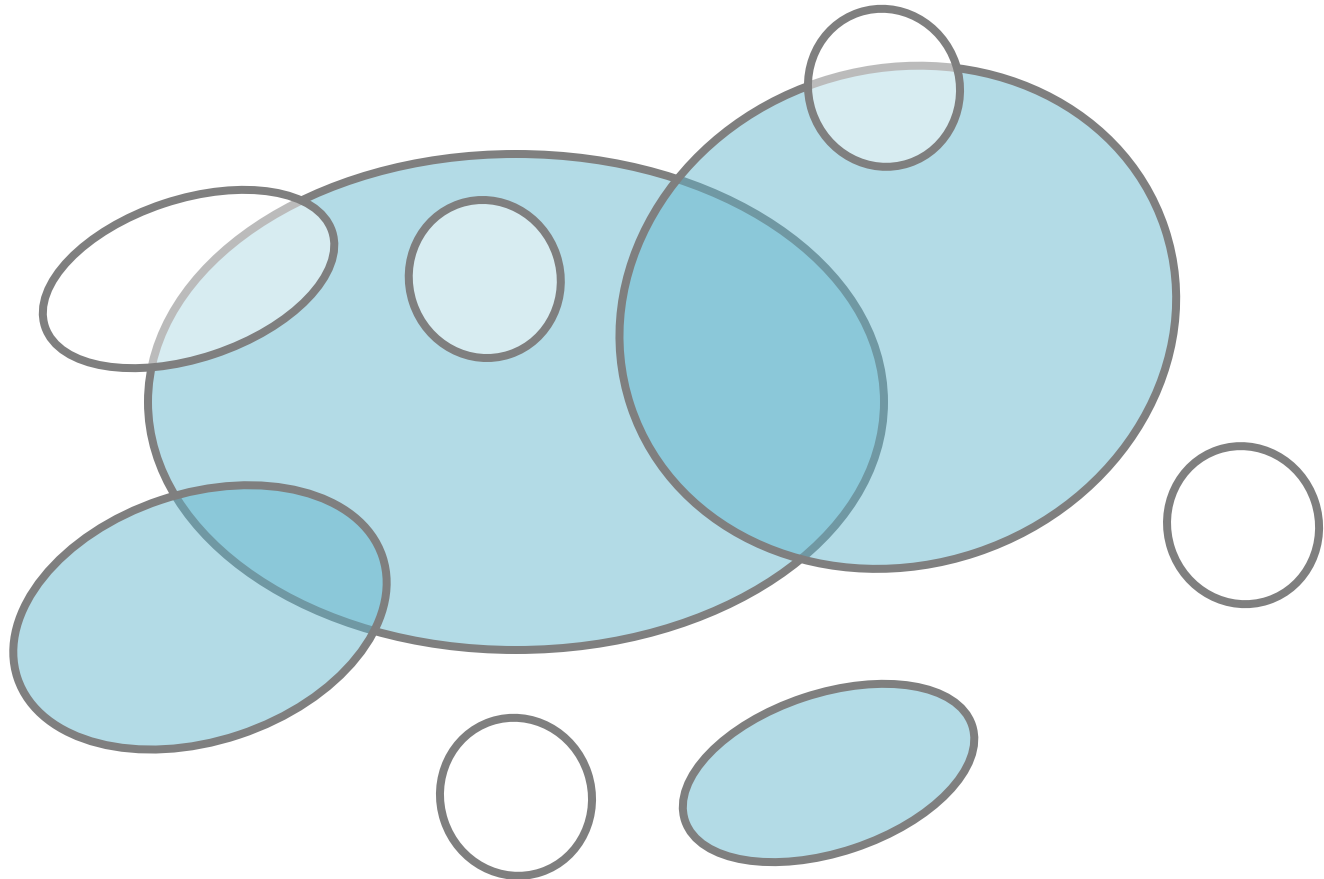
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**

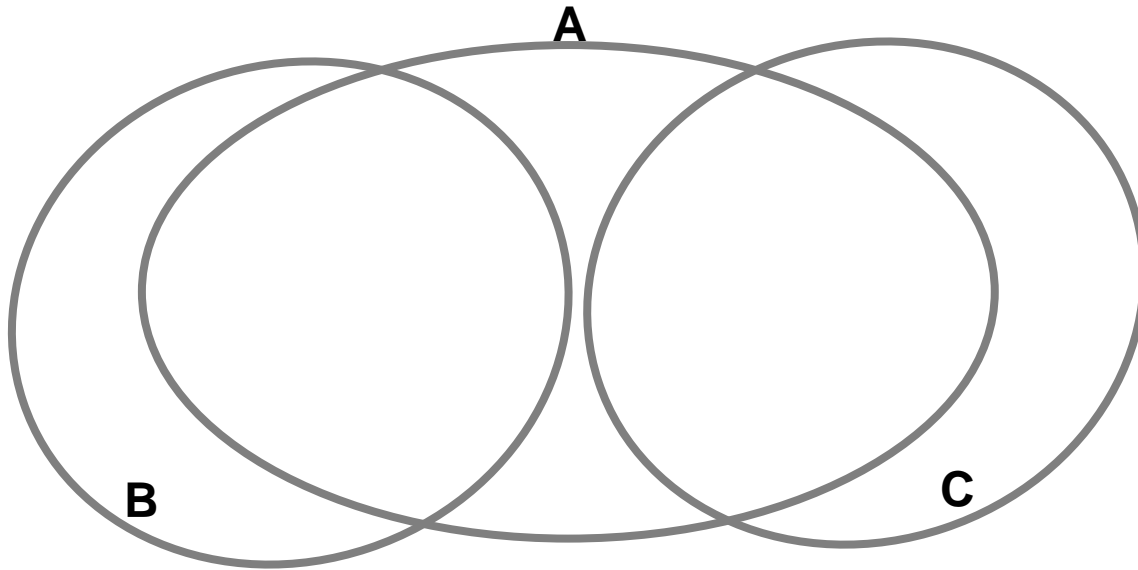


# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



# When Greedy Heuristic Fails?



- Goal: Maximize the size of the covered area
- Greedy first picks A and then C
- But the optimal way would be to pick B and C

# Approximation Guarantee

- **Greedy produces a solution  $A$**   
**where:  $F(A) \geq (1-1/e)*OPT$  ( $F(A) > 0.63*OPT$ )**  
[Nemhauser, Fisher, Wolsey '78]
- **Claim holds for functions  $F(\cdot)$  with 2 properties:**
  - **$F$  is monotone:** (adding more docs doesn't decrease coverage)  
if  $A \subseteq B$  then  $F(A) \leq F(B)$  and  $F(\{\})=0$
  - **$F$  is submodular:**  
adding an element to a set gives less improvement  
than adding it to one of its subsets



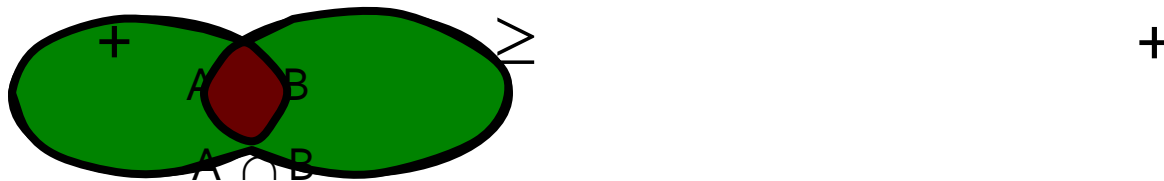
# Submodularity: Definition

## Definition:

- Set function  $F(\cdot)$  is called **submodular** if:

For all  $A, B \subseteq W$ :

$$F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$$



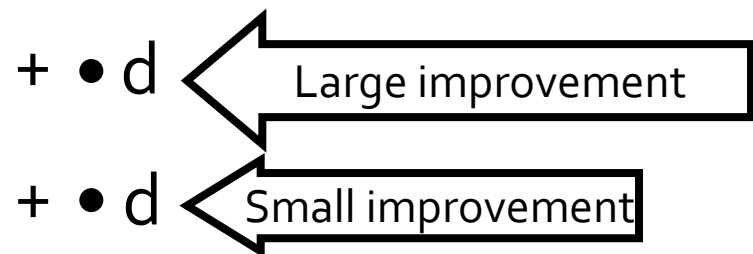
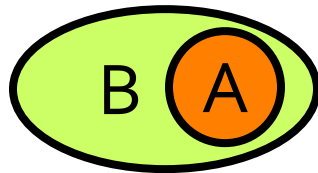
# Submodularity: Or equivalently

- **Diminishing returns** characterization

**Equivalent definition:**

- Set function  $F(\cdot)$  is called **submodular** if:  
For all  $A \subseteq B$ ,  $s \notin B$ :

$$\underbrace{F(A \cup d) - F(A)}_{\text{Gain of adding } d \text{ to a small set}} \geq \underbrace{F(B \cup d) - F(B)}_{\text{Gain of adding } d \text{ to a large set}}$$



# Example: Set Cover

- $F(\cdot)$  is **submodular**:  $A \subseteq B$

$$\underbrace{F(A \cup d) - F(A)}_{\text{Gain of adding } X_d \text{ to a small set}} \geq \underbrace{F(B \cup d) - F(B)}_{\text{Gain of adding } X_d \text{ to a large set}}$$

Gain of adding  $X_d$  to a small set

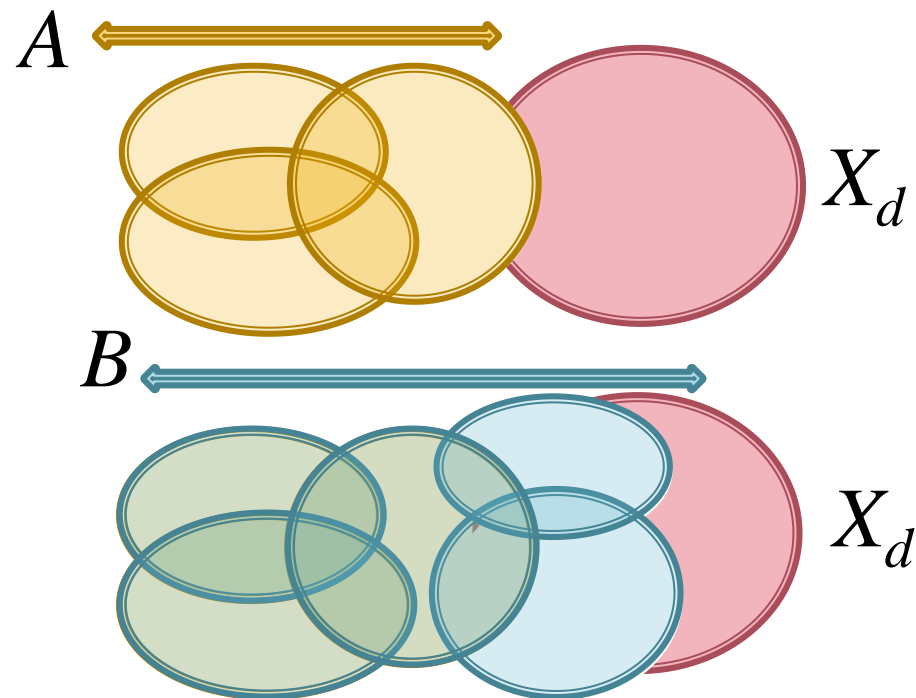
Gain of adding  $X_d$  to a large set

- **Natural example:**

- Sets  $X_1, \dots, X_m$
- $F(A) = |\cup_{d \in A} X_d|$   
(size of the covered area)

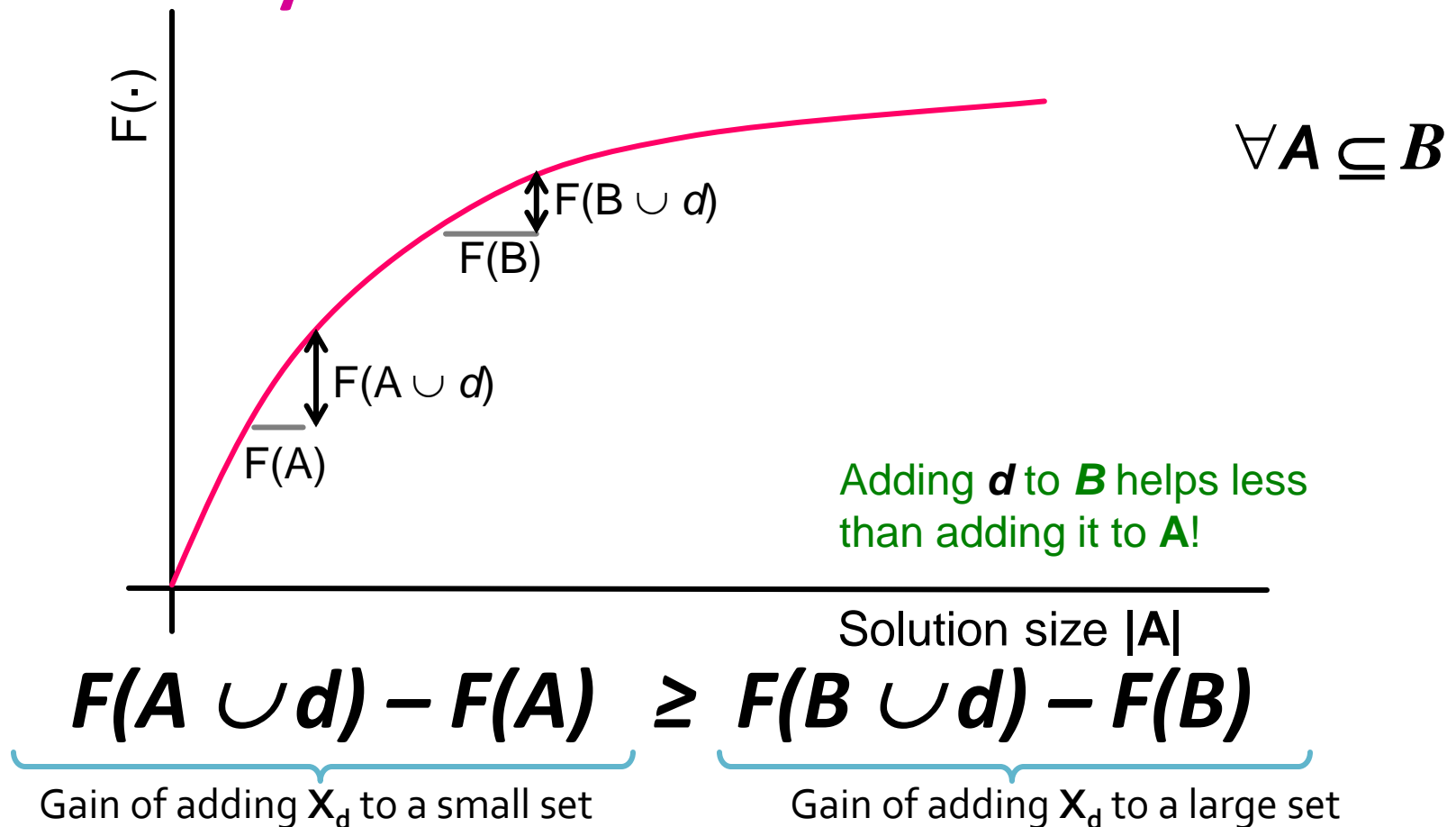
- **Claim:**

$F(A)$  is submodular!



# Submodularity– Diminishing returns

- Submodularity is discrete analogue of concavity



# Submodularity & Concavity

- **Marginal gain:**

$$\Delta_F(d|A) = F(A \cup X_d) - F(A)$$

- **Submodular:**

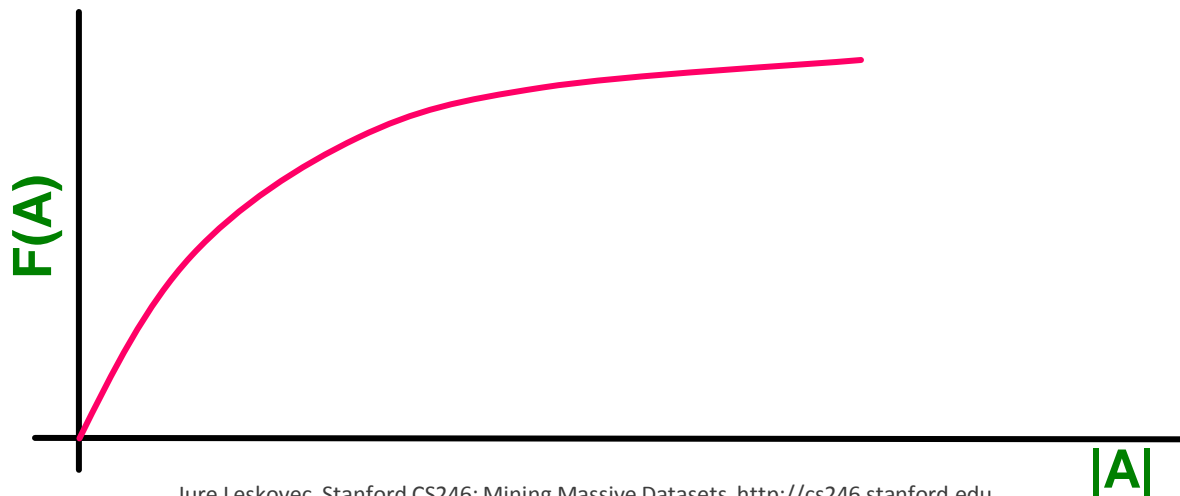
$$A \subseteq B$$

$$F(A \cup d) - F(A) \geq F(B \cup d) - F(B)$$

- **Concavity:**

$$a \leq b$$

$$f(a + d) - f(a) \geq f(b + d) - f(b)$$

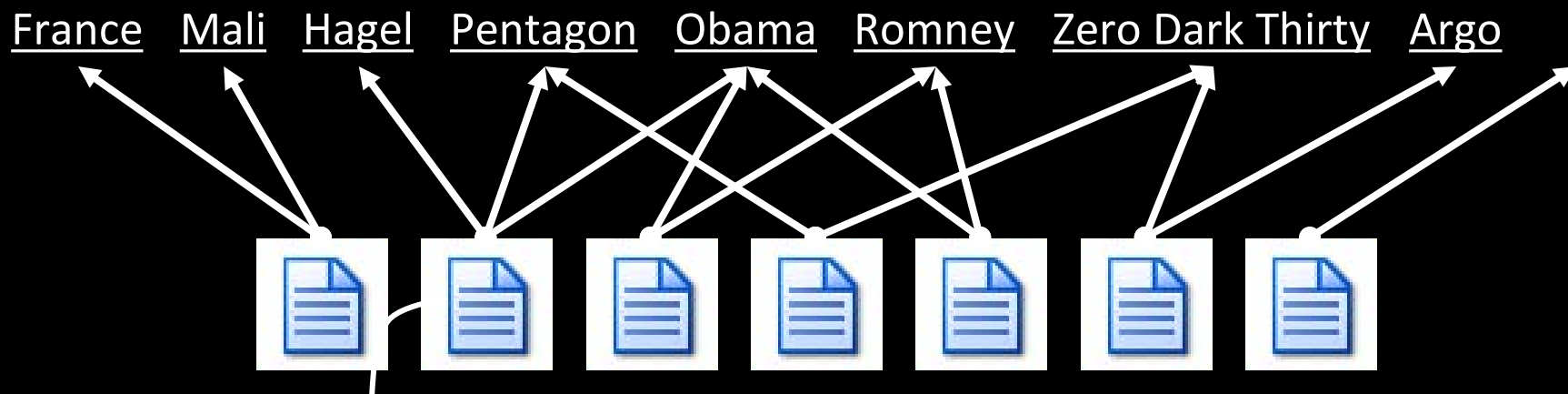


# Submodularity: Useful Fact

- Let  $F_1 \dots F_m$  be **submodular** and  $\lambda_1 \dots \lambda_m > 0$   
then  $F(A) = \sum_i^m \lambda_i F_i(A)$  is **submodular**
  - **Submodularity is closed under non-negative linear combinations!**
  - **This is an extremely useful fact:**
    - **Average of submodular functions is submodular:**  
 $F(A) = \sum_i P(i) \cdot F_i(A)$
    - **Multicriterion optimization:**  $F(A) = \sum_i \lambda_i F_i(A)$

# Back to our problem

- **Q: What is being covered?**
- **A: Concepts** (In our case: Named entities)

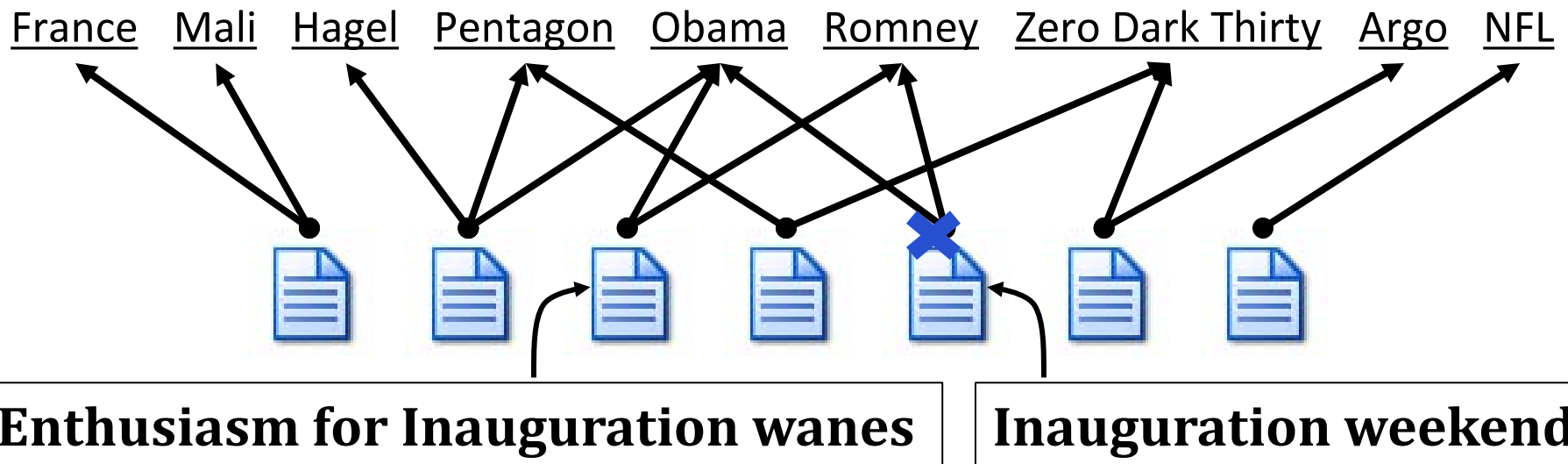


Hagel expects fight

- **Q: Who is doing the covering?**
- **A: Documents**

# Back to our Concept Cover Problem

- **Objective:** pick  $k$  docs that cover most concepts

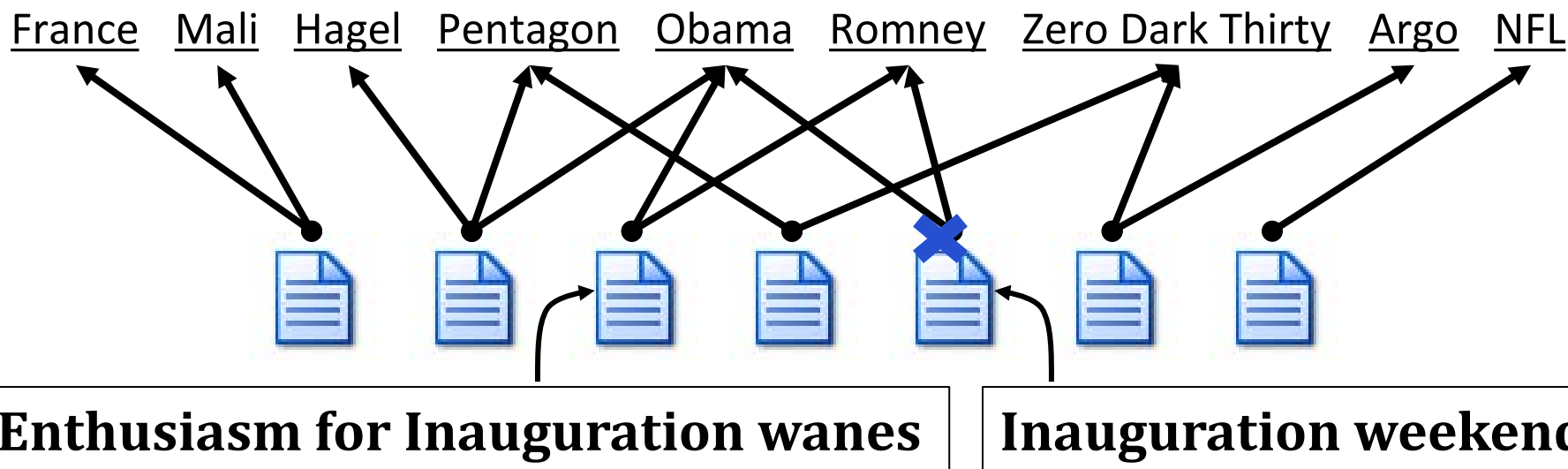


- $F(A)$ : the number of concepts covered by  $A$ 
  - *Elements...concepts, Sets ... concepts in docs*
  - $F(A)$  is submodular and monotone!
  - We can use **greedy** to optimize  $F$



# The Set Cover Problem

- **Objective:** pick  $k$  docs that cover most concepts



The good:

Penalizes redundancy

Submodular

The bad:

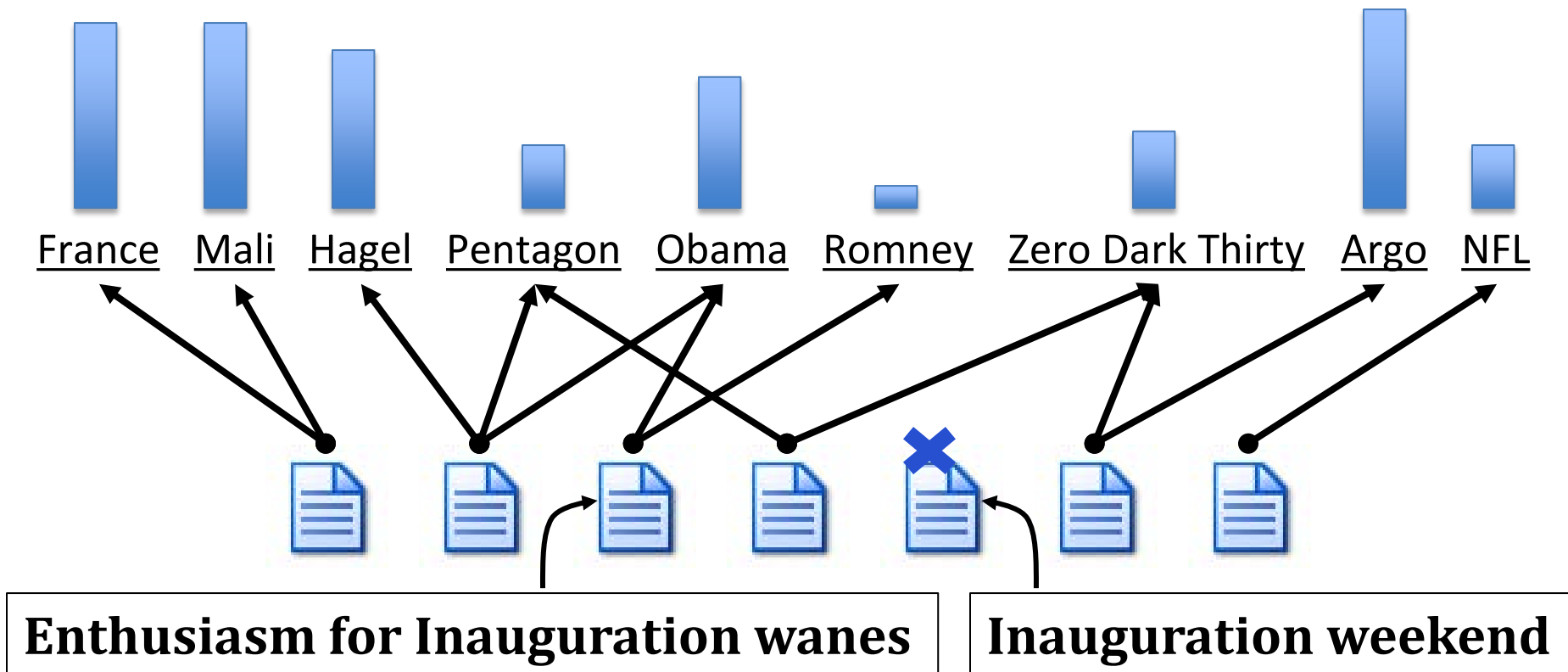
Concept importance?

All-or-nothing too harsh

# Probabilistic Set Cover

# Concept importance?

- **Objective:** pick  $k$  docs that cover most concepts

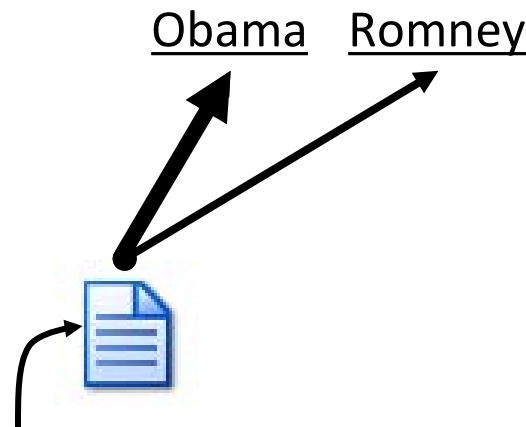


- Each concept  $c$  has importance weight  $w_c$

# All-or-nothing too harsh

- **Document coverage function**

$\text{cover}_d(c)$  = **probability** document **d** covers  
concept **c**  
[e.g., how strongly **d** covers **c**]



**Enthusiasm for Inauguration wanes**

# Probabilistic Set Cover

- **Document coverage function:**

$\text{cover}_d(c) = \text{probability document } d \text{ covers concept } c$

- $\text{Cover}_d(c)$  can model how relevant is concept  $c$  for user  $u$

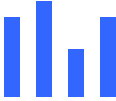
- **Set coverage function:**

$$\text{cover}_{\mathcal{A}}(c) = 1 - \prod_{d \in \mathcal{A}} (1 - \text{cover}_d(c))$$

- Prob. that at least one document in  $\mathbf{A}$  covers  $\mathbf{c}$

- **Objective:**

$$\max_{\mathcal{A}: |\mathcal{A}| \leq k} F(\mathcal{A}) = \sum_c w_c \text{cover}_{\mathcal{A}}(c)$$

concept weights 

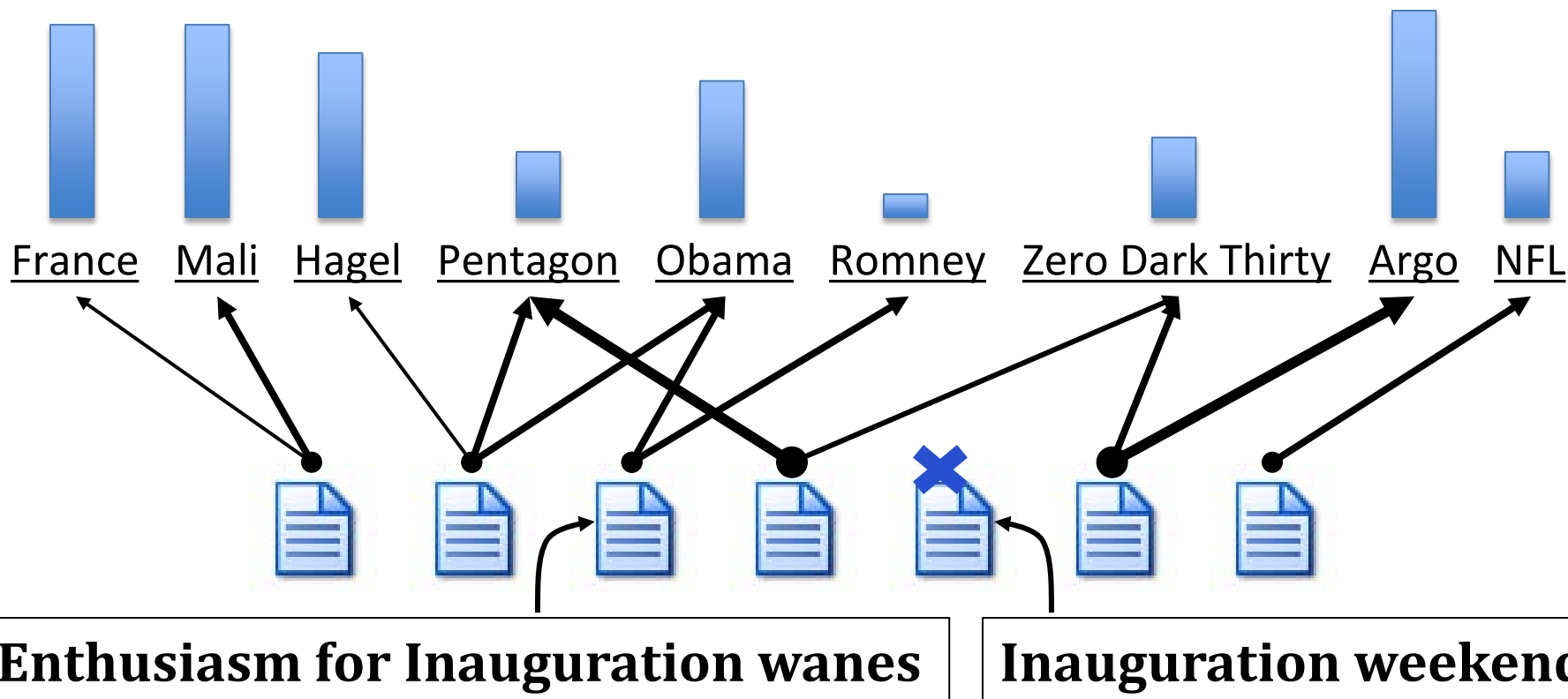
# Optimizing $F(\mathcal{A})$

$$\max_{\mathcal{A}: |\mathcal{A}| \leq k} F(\mathcal{A}) = \sum_c w_c \text{cover}_{\mathcal{A}}(c)$$

- The objective function is also **submodular**
  - Intuitive **diminishing returns** property
  - Greedy algorithm leads to a  $(1 - 1/e) \sim 63\%$  approximation, i.e., a **near-optimal** solution

# Summary: Probabilistic Set Cover

- **Objective:** pick  $k$  docs that cover most concepts



- Each concept  $c$  has importance weight  $w_c$
- Documents partially cover concepts:  $\text{cover}_d(c)$

# Lazy Optimization of Submodular Functions

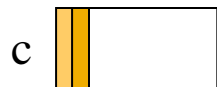


# Submodular Functions

## Greedy

Marginal gain:

$$F(A \cup x) - F(A)$$



Add document with  
highest marginal gain

## ■ Greedy algorithm is slow!

- At each iteration we need to re-evaluate marginal gains of **all remaining documents**
- Runtime  $O(|V| \cdot K)$  for selecting  $K$  documents

# Speeding up Greedy

- **In round  $i$ :** So far we have  $A_{i-1} = \{d_1, \dots, d_{i-1}\}$ 
  - Now we pick  $d_i = \arg \max_{d \in V} F(A_{i-1} \cup \{d\}) - F(A_{i-1})$ 
    - Greedy algorithm maximizes the “marginal benefit”  

$$\Delta_i(d) = F(A_{i-1} \cup \{d\}) - F(A_{i-1})$$

- **By submodularity property:**

$$F(A_i \cup \{d\}) - F(A_i) \geq F(A_j \cup \{d\}) - F(A_j) \text{ for } i < j$$

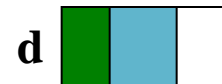
- **Observation: By submodularity:**

**For every  $d \in V$**

$$\Delta_i(d) \geq \Delta_j(d) \text{ for } i < j \text{ since } A_i \subseteq A_j$$

$$\Delta_i(d) \geq \Delta_j(d)$$

- **Marginal benefits  $\Delta_i(d)$  only shrink!**  
**(as  $i$  grows)**



Selecting document  $d$  in step  $i$  covers more words than selecting  $d$  at step  $j$  ( $j > i$ )

# Lazy Greedy

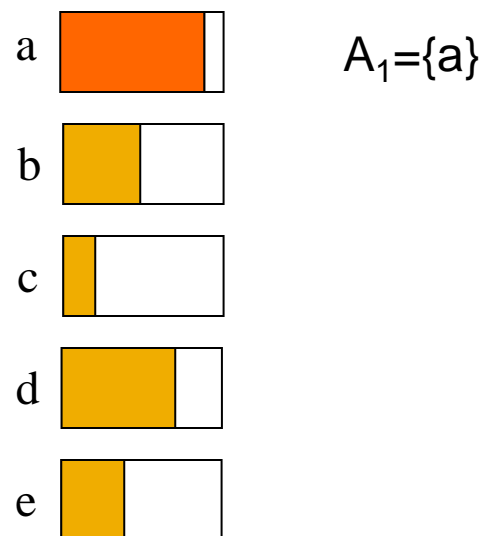
## ■ Idea:

- Use  $\Delta_i$  as upper-bound on  $\Delta_j$  ( $j > i$ )

## ■ Lazy Greedy:

- Keep an ordered list of marginal benefits  $\Delta_j$  from previous iteration
- Re-evaluate  $\Delta_j$  **only** for top node
- Re-sort and prune

Marginal gain



$$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad A \subseteq B$$

# Lazy Greedy

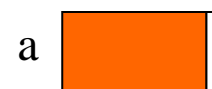
## ■ Idea:

- Use  $\Delta_i$  as upper-bound on  $\Delta_j$  ( $j > i$ )

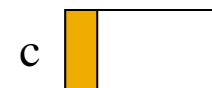
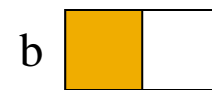
## ■ Lazy Greedy:

- Keep an ordered list of marginal benefits  $\Delta_j$  from previous iteration
- Re-evaluate  $\Delta_j$  **only** for top node
- Re-sort and prune

Marginal gain



$A_1 = \{a\}$



$$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad A \subseteq B$$

# Lazy Greedy

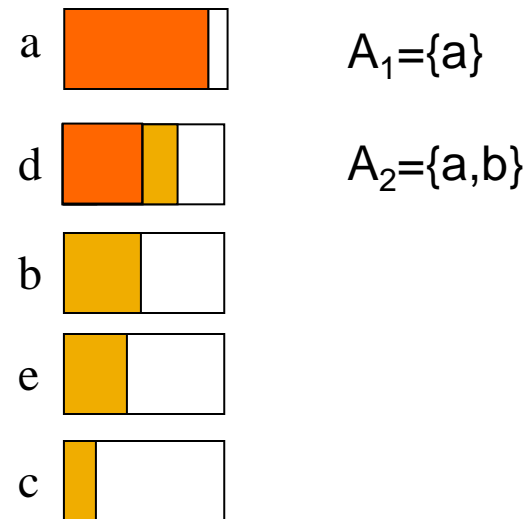
## ■ Idea:

- Use  $\Delta_i$  as upper-bound on  $\Delta_j$  ( $j > i$ )

## ■ Lazy Greedy:

- Keep an ordered list of marginal benefits  $\Delta_j$  from previous iteration
- Re-evaluate  $\Delta_j$  **only** for top node
- Re-sort and prune

Marginal gain

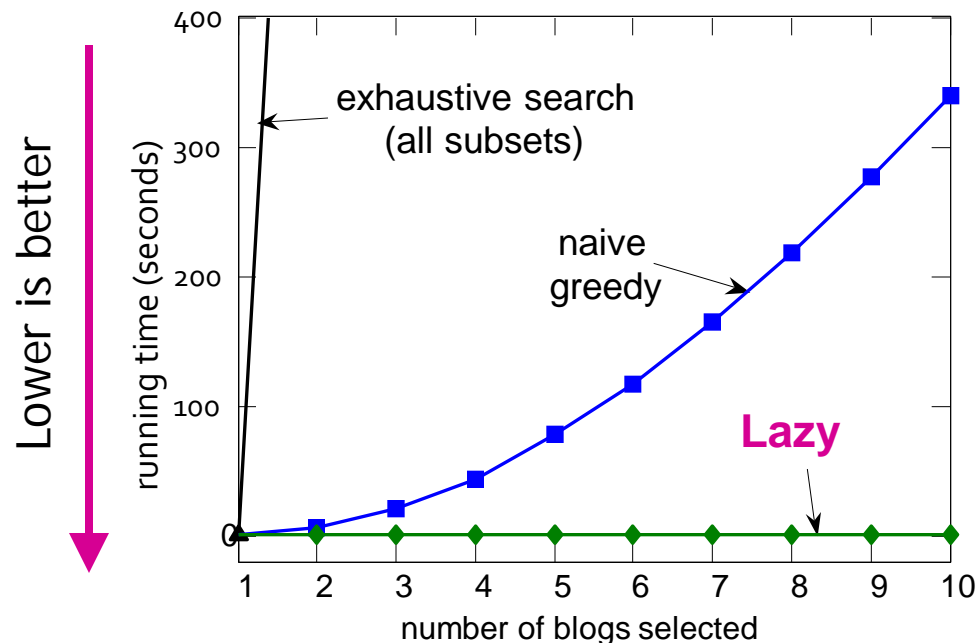


$$F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B) \quad A \subseteq B$$

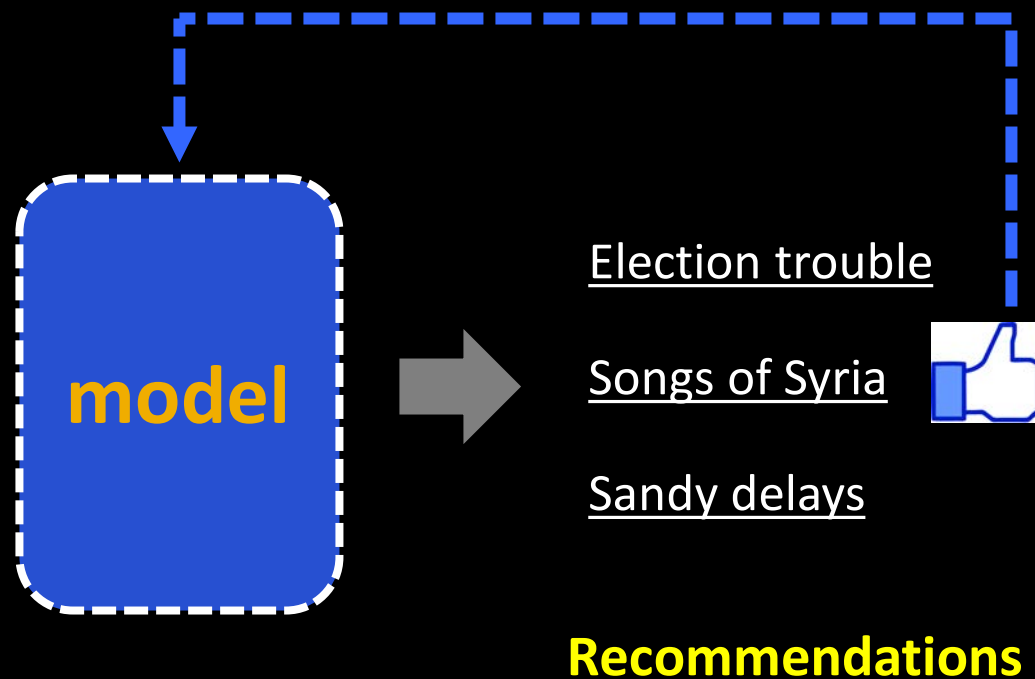
# Summary so far

## ■ Summary so far:

- Diversity can be formulated as a set cover
- Set cover is submodular optimization problem
- Can be (approximately) solved using greedy algorithm
- Lazy-greedy gives significant speedup

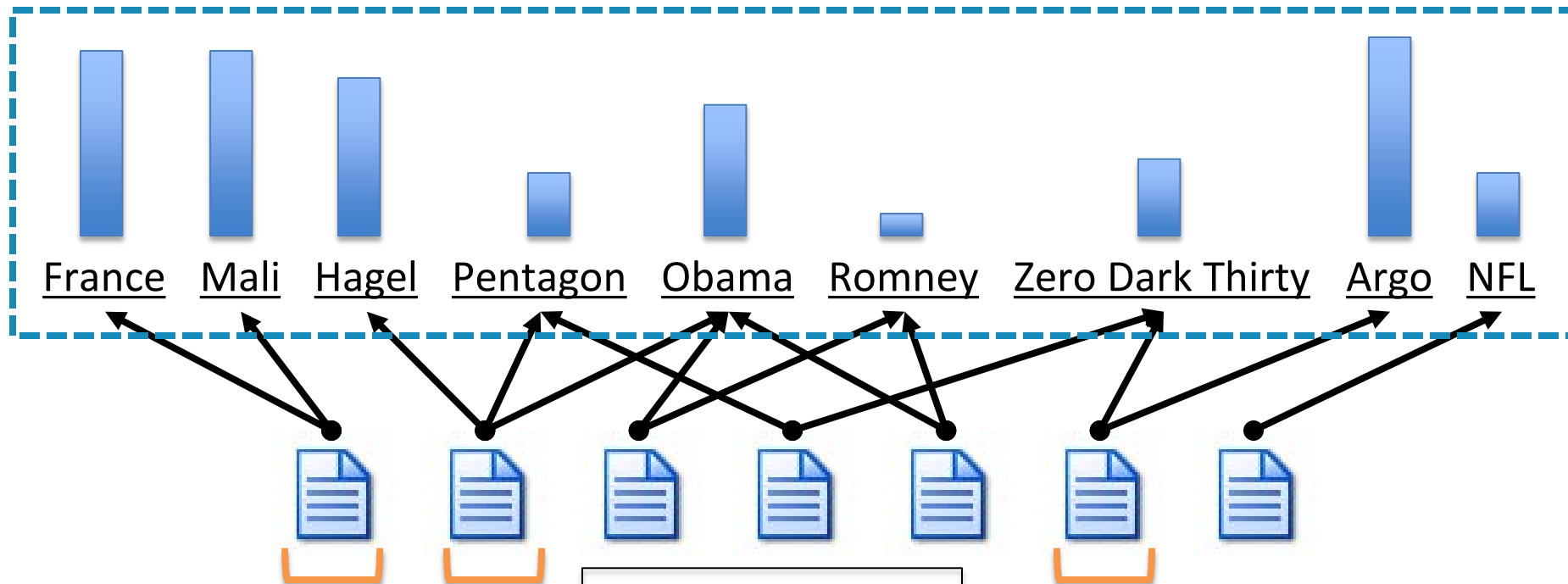


# But what about **personalization?**



# Concept Coverage

We assumed same concept weighting for all users



France intervenes

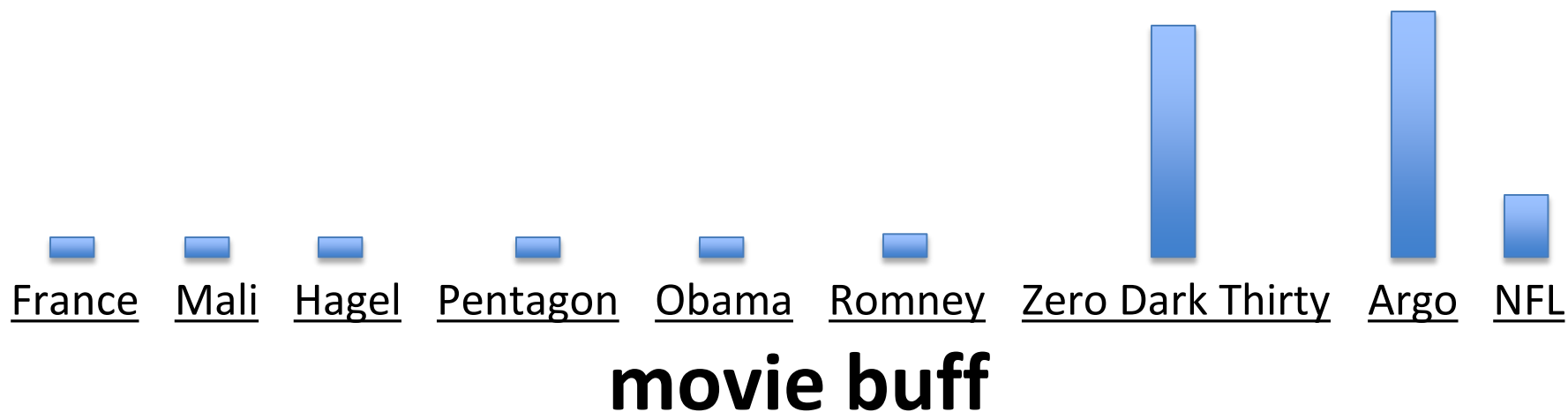
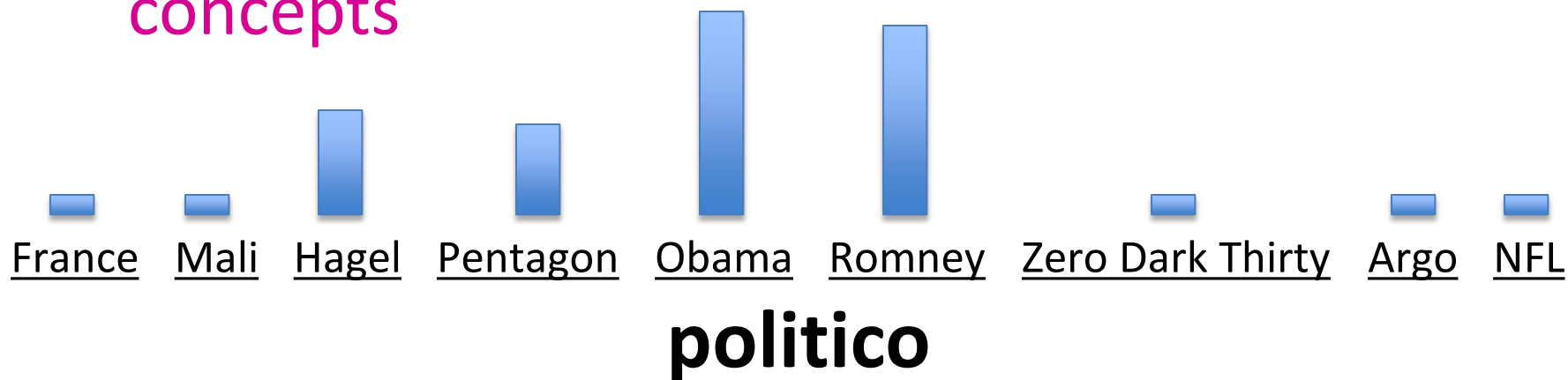
Chuck for Defense

Argo wins big



# Personal Concept Weights

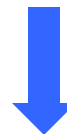
- Each user has **different** preferences over concepts



# Personal concept weights

- Assume each user has **different** preference vector over concepts

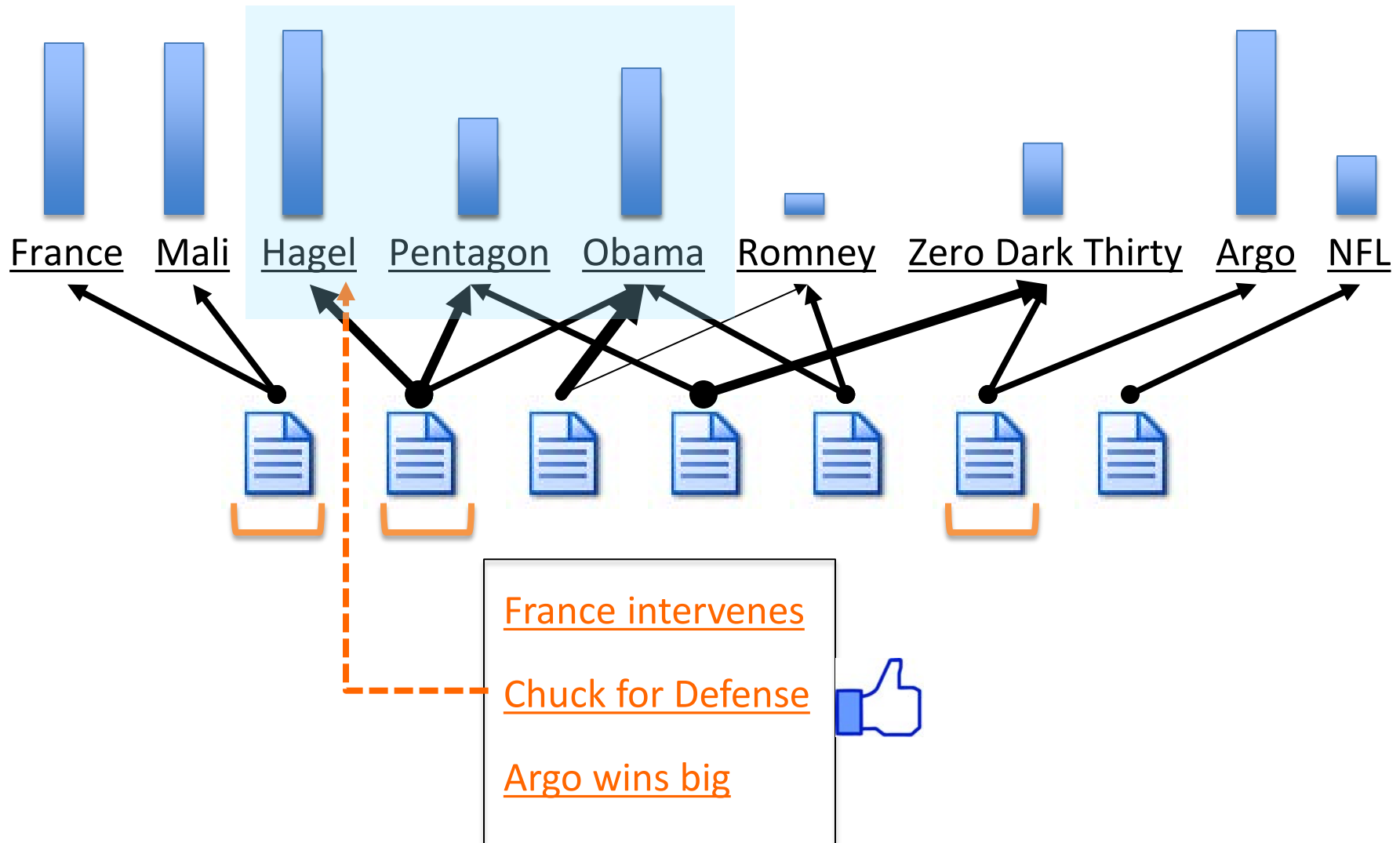
$$\max_{\mathcal{A}: |\mathcal{A}| \leq k} F(\mathcal{A}) = \sum_c w_c \text{cover}_{\mathcal{A}}(c)$$



$$\max_{\mathcal{A}: |\mathcal{A}| \leq k} F(\mathcal{A}) = \sum_c w_c^{(u)} \text{cover}_{\mathcal{A}}(c)$$

- Goal:** Learn personal concept weights from user feedback

# Interactive Concept Coverage



# Multiplicative Weights (MW)

- **Multiplicative Weights algorithm**
  - Assume each concept  $c$  has weight  $w_c$
  - We recommend document  $d$  and receive feedback, say  $r = +1$  or  $-1$
  - **Update the weights:**
    - If  $c \in X_d$  then  $w_c = \beta^r w_c$
    - If  $c \notin X_d$  then  $w_c = \beta^{-r} w_c$ 
      - If concept  $c$  appears in  $X_d$  and we received positive feedback  $r=+1$  then we increase the weight  $w_c$  by multiplying it by  $\beta$  ( $\beta > 1$ ) otherwise we decrease the weight (divide by  $\beta$ )
  - **Normalize weights so that  $\sum_c w_c = 1$**

# Summary of the Algorithm

## ■ Steps of the algorithm:

1. Identify **items** to recommend from
2. Identify **concepts** [what makes items redundant?]
3. **Weigh** concepts by general importance
4. Define **item-concept coverage function**
5. **Select** items using probabilistic set cover
6. Obtain **feedback**, **update** weights

# Summary: Submodularity

	Maximization	Minimization
Unconstrained	NP-hard, but well-approximable (if nonnegative)	Polynomial time! Generally inefficient ( $n^6$ ), but can exploit special cases (cuts; symmetry; decomposable; ...)
Constrained	NP-hard but well-approximable „Greedy-(like)“ for cardinality, matroid constraints; Non-greedy for more complex (e.g., connectivity) constraints	NP-hard; hard to approximate, still useful algorithms