

- Announcements
  - Colab 9 on causal inference – out today, due next Sun
  - Lecture next week on causal inference – be ready to do the colab

# Submodular Optimization

---

CSEP590A Machine Learning for Big Data

Tim Althoff



PAUL G. ALLEN SCHOOL  
OF COMPUTER SCIENCE & ENGINEERING

# Motivation

- Learned about: LSH/Similarity search & recommender systems

- Search: “jaguar”



- **Uncertainty** about the user's information need
  - Don't put all eggs in one basket!
- **Relevance** isn't everything – need **diversity**!

# Many applications need diversity!

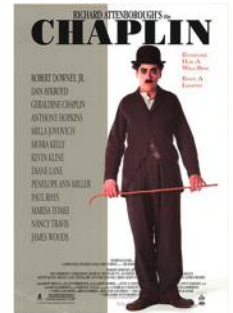
- Recommendation:

NETFLIX

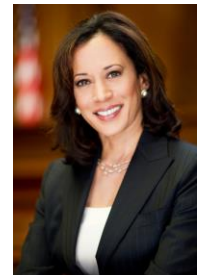


- Summarization:  
“Robert Downey Jr.”

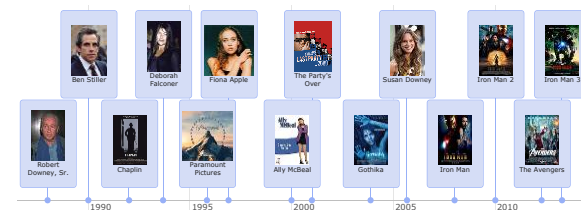
WIKIPEDIA



- News Media:



# Automatic Timeline Generation



## Person

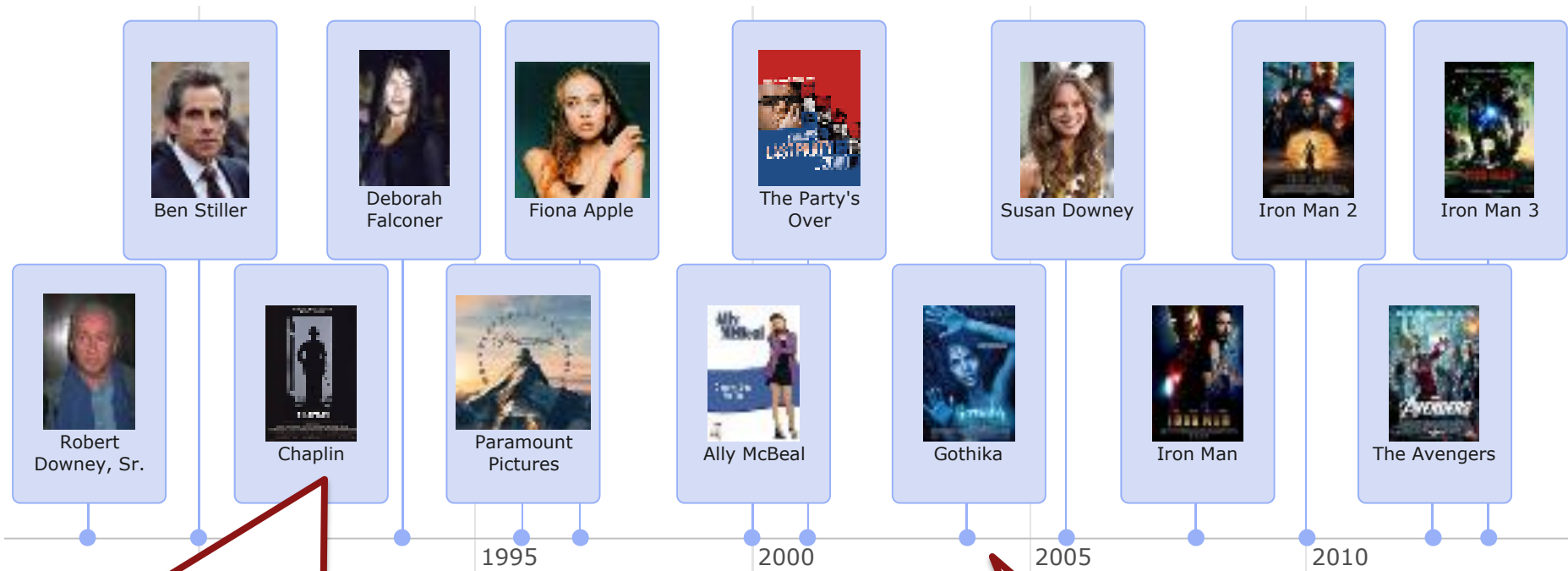
## Timeline

- **Goal:** Timeline should express their **relationships** to other people through **events** (personal, collaboration, mentorship, etc.)
- **Why timelines?**
  - Easier: Wikipedia article is 18 pages long
  - Context: Through relationships & event descriptions
  - Exploration: Can “jump” to other people/entities

# Problem Definition

- **Given:**
  - Relevant **relationships**
  - **Events** that each cover some relationships
- **Goal:** Given a **large set of events**, pick a **small subset** that explains most known **relationships** (“the timeline”)

# Example Timeline



"RDJr starred in Chaplin in 1992 together with Anthony Hopkins."

Good overview

# Why diversity?

- User studies: People hate redundancy!

**Iron Man**  
US Release

**Iron Man**  
Award  
Ceremony

**VS**

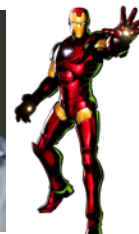
**Iron Man**  
US Release

**Chaplin**  
Academy  
Award N.

**Iron Man**  
EU Release

**Rented Lips**  
US Release

- Want to see more diverse set of relationships



# Diversity as Coverage

---



# Encode Diversity as Coverage

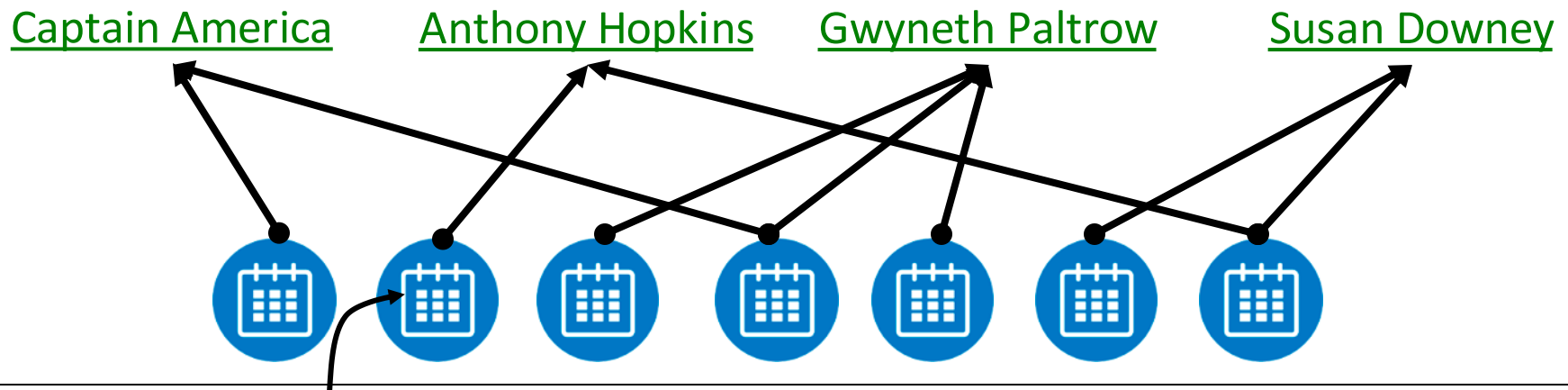
- **Idea:** Encode diversity as coverage problem
- **Example:** Selecting events for timeline
  - Try to cover all important relationships



# What is being covered?

- Q: What is being covered?

- A: Relationships



Downey Jr. starred in *Chaplin* together with Anthony Hopkins

- Q: Who is doing the covering?

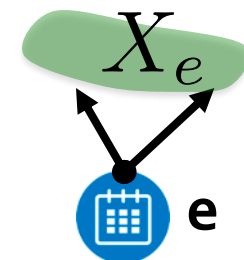
- A: Timeline Events

# Simple Coverage Model

- Suppose we are given a set of events  $E$ 
  - Each event  $e$  covers a set  $X_e \subseteq U$  of relationships
- For a set of events  $S \subseteq E$  we define:

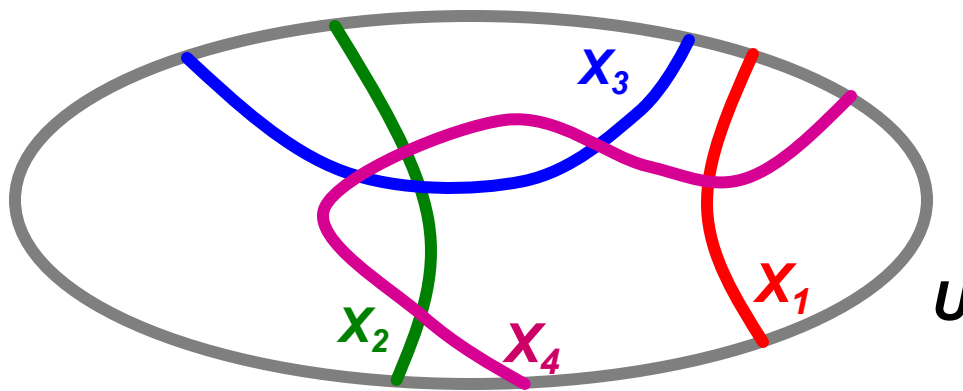
$$F(S) = \left| \bigcup_{e \in S} X_e \right|$$

- Goal: We want to  $\max_{|S| \leq k} F(S)$  ← Cardinality Constraint
- Note:  $F(S)$  is a set function:  $F(S) : 2^E \rightarrow \mathbb{N}$



# Maximum Coverage Problem

- **Given universe of elements**  $U = \{u_1, \dots, u_n\}$   
**and sets**  $\{X_1, \dots, X_m\} \subseteq U$



$U$ : all relationships  
 $X_i$ : relationships covered by event  $i$

- **Goal: Find set of  $k$  events  $X_1 \dots X_k$  covering most of  $U$** 
  - More precisely: Find set of  $k$  events  $X_1 \dots X_k$  whose size of the union is the largest

# Simple Greedy Heuristic

## Simple Heuristic: Greedy Algorithm:

- Start with  $S_0 = \{\}$
- For  $i = 1 \dots k$ 
  - Take event  $e$  that  $\max F(S_{i-1} \cup e)$
  - Let  $S_i = S_{i-1} \cup \{e\}$

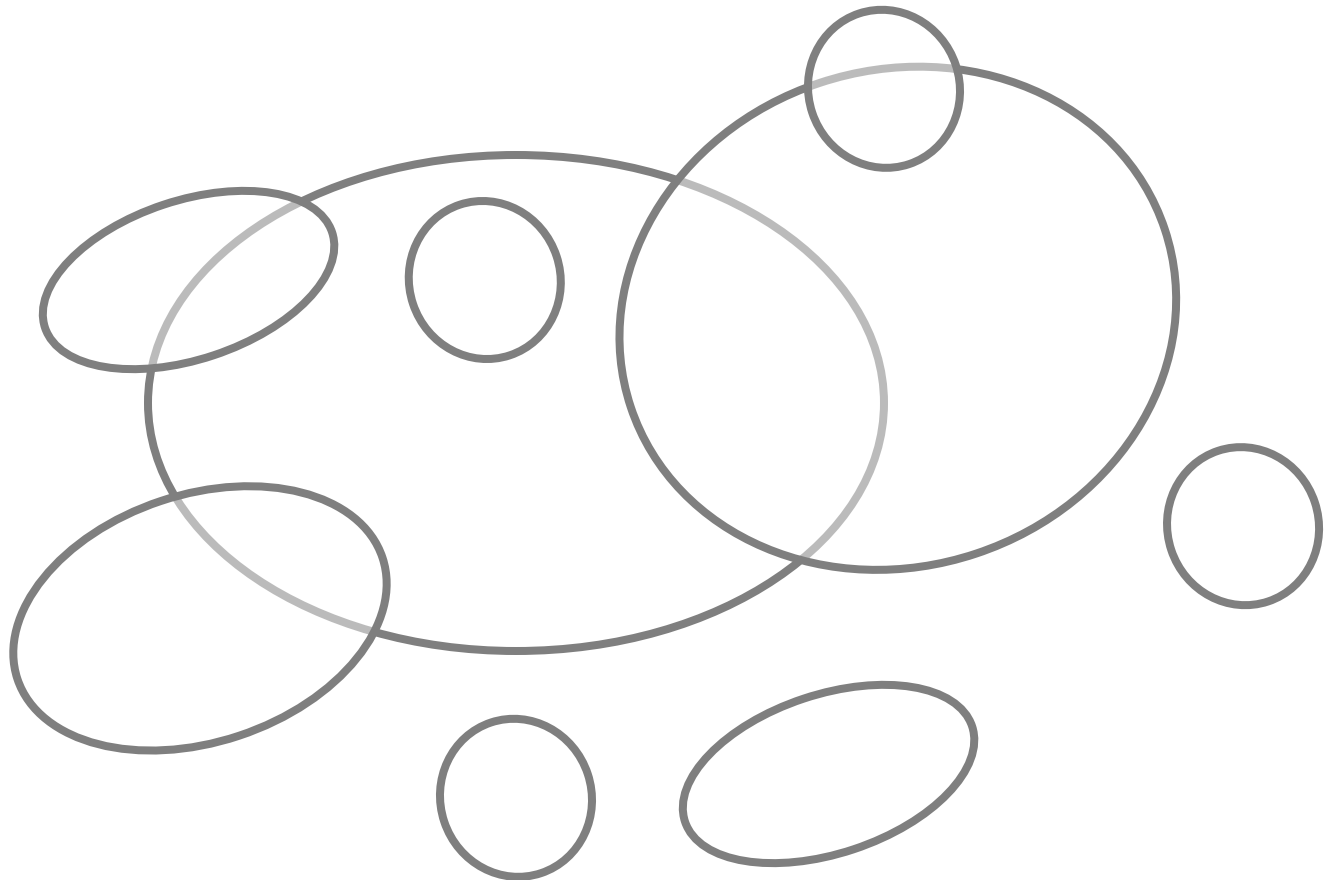
$$F(S) = \left| \bigcup_{e \in S} X_e \right|$$

## ■ Example:

- Eval.  $F(\{e_1\})$ , ...,  $F(\{e_m\})$ , pick best (say  $e_1$ )
- Eval.  $F(\{e_1\} \cup \{e_2\})$ , ...,  $F(\{e_1\} \cup \{e_m\})$ , pick best (say  $e_2$ )
- Eval.  $F(\{e_1, e_2\} \cup \{e_3\})$ , ...,  $F(\{e_1, e_2\} \cup \{e_m\})$ , pick best
- And so on...

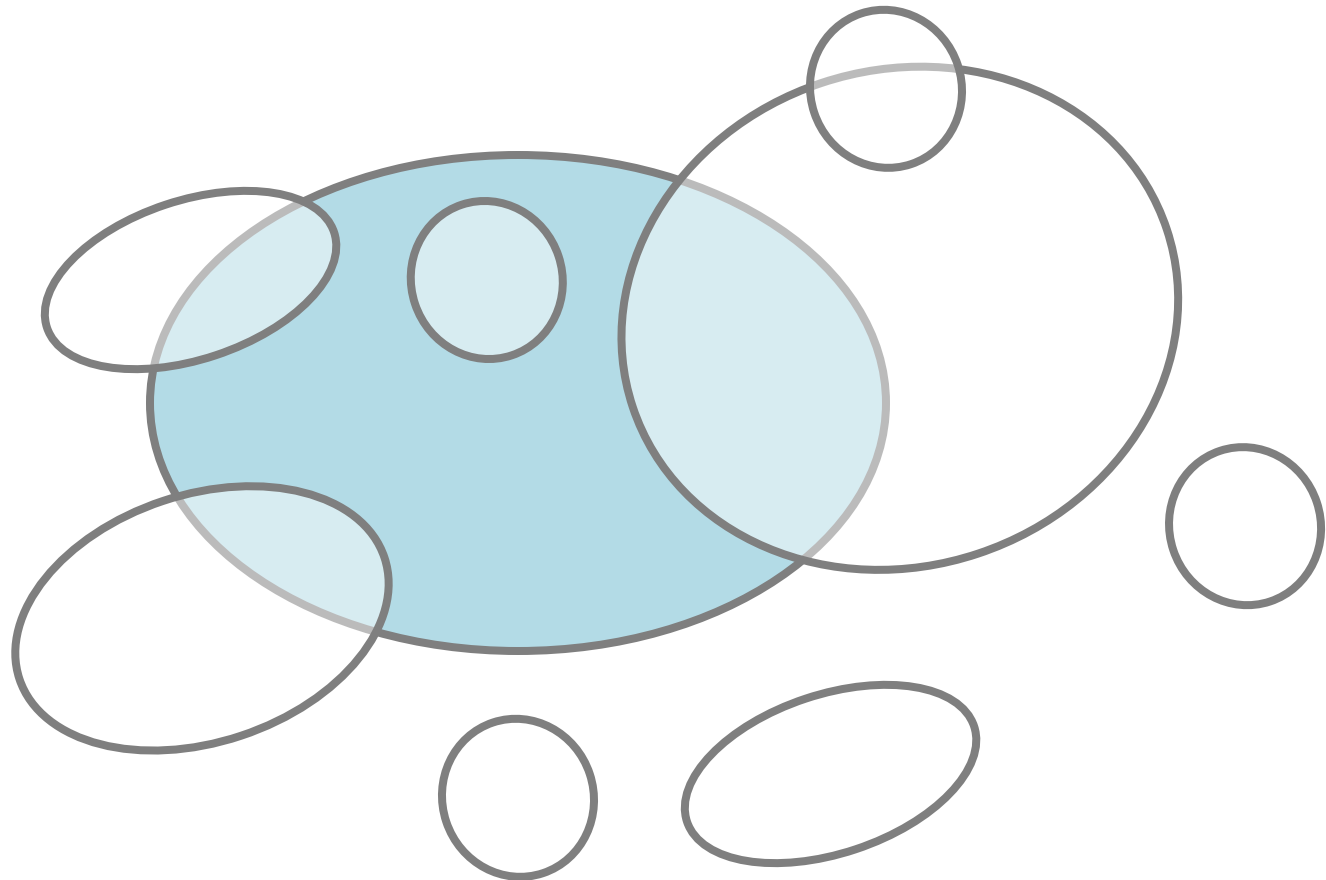
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



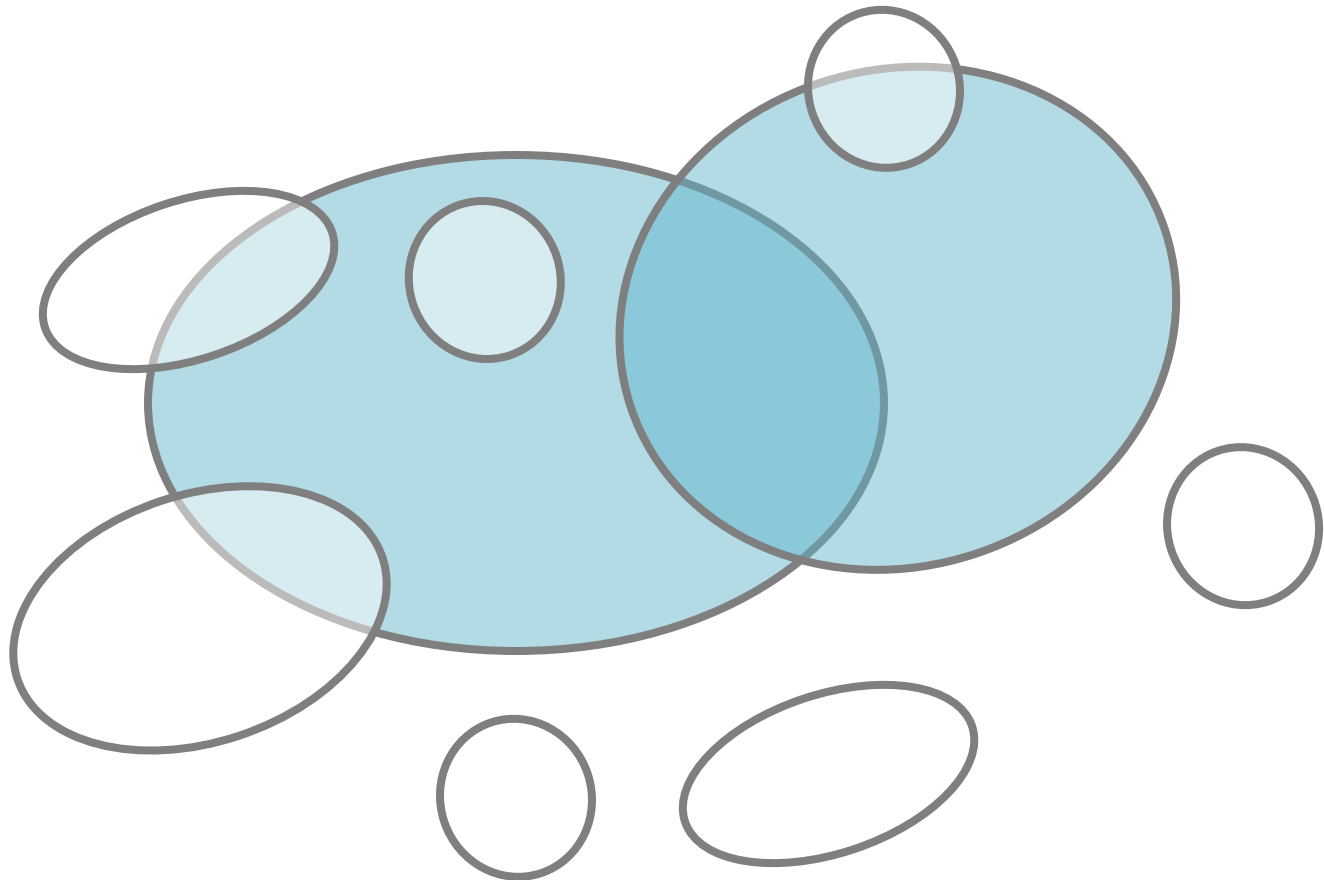
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



# Simple Greedy Heuristic

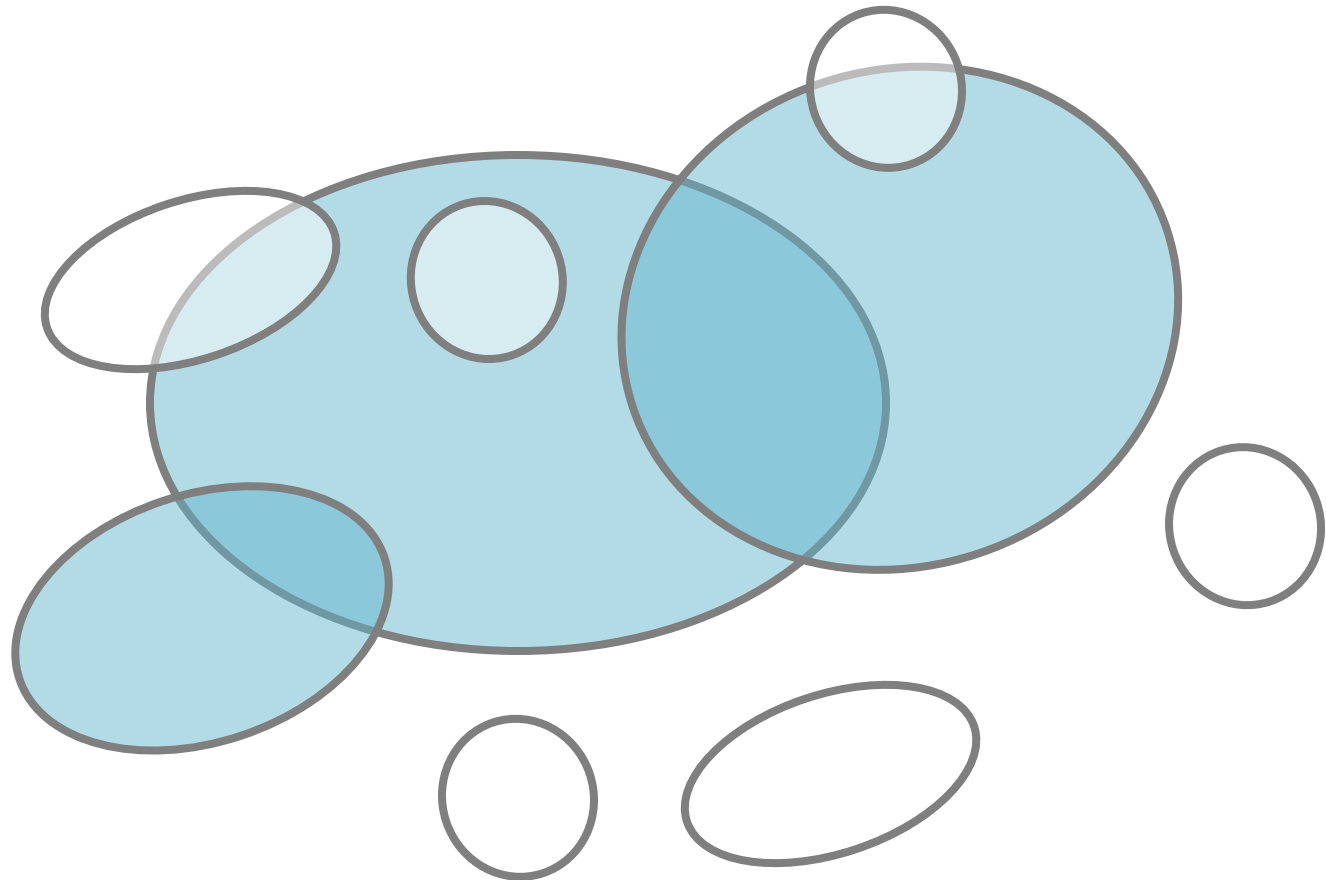
- **Goal: Maximize the covered area**





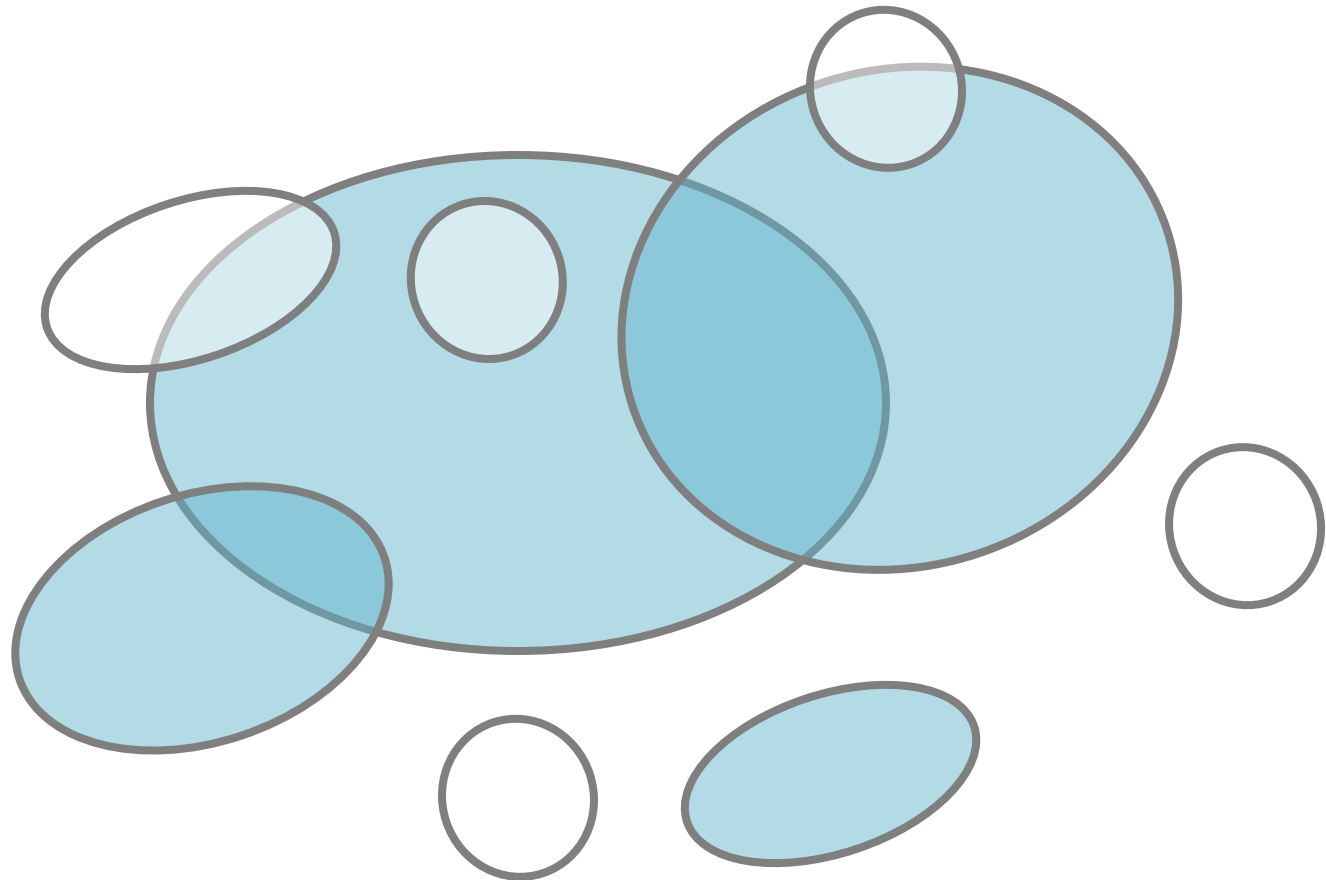
# Simple Greedy Heuristic

- **Goal: Maximize the covered area**

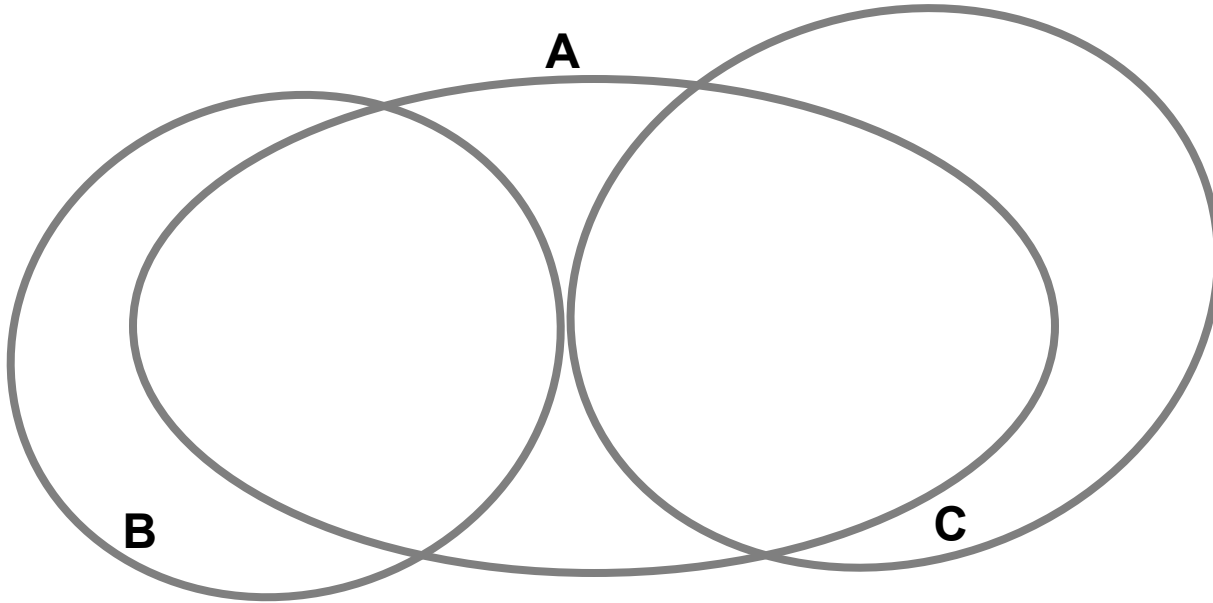


# Simple Greedy Heuristic

- **Goal: Maximize the covered area**



# When Greedy Heuristic Fails?



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

# Bad News & Good News

- **Bad news:** Maximum Coverage is **NP-hard**
  - Related to Set Cover Problem
- **Good news:** Good **approximations** exist
  - Problem has certain **structure** to it that even simple greedy algorithms perform reasonably well
  - Details in 2<sup>nd</sup> half of lecture
- **Now: Generalize** our objective for timeline generation

# Issue 1: Not all relationships are created equal

- Objective values all relationships **equally**

$$F(S) = \left| \bigcup_{e \in S} X_e \right| = \sum_{r \in R} 1 \text{ where } R = \bigcup_{e \in S} X_e$$

- **Unrealistic:** Some relationships are more important than others
  - use **different weights** (“weighted coverage function”)

$$F(S) = \sum_{r \in R} w(r) \quad w : R \rightarrow \mathbb{R}^+$$

# Example weight function

- Use **global importance** weights
- How much interest is there?
- Could be measured as
  - $w(X)$  = # **search queries** for person X
  - $w(X)$  = # **Wikipedia article views** for X
  - $w(X)$  = # **news article mentions** for X

Captain America

Anthony Hopkins

Gwyneth Paltrow

Susan Downey



Captain America Anthony Hopkins Gwyneth Paltrow Susan Downey

# Better weight function

Captain America

Justin Bieber

Susan Downey

Tim Althoff



Applying global importance weights

Captain America

Justin Bieber

Susan Downey

Tim Althoff

- Some relationships are **not (very) globally important but (not) highly relevant** to timeline
- Need **relevant to timeline** instead of **globally relevant**

$$w(\text{Susan Downey} \mid \text{RDJr}) > w(\text{Justin Bieber} \mid \text{RDJr})$$

# Capturing relevance to timeline

- Can use co-occurrence statistics

$$w(X \mid \text{RDJr}) = \#(X \text{ and RDJr}) / (\#(\text{RDJr}) * \#(X))$$

- Similar: Pointwise mutual information (PMI)
- How often do X and Y occur together compared to what you would expect if they were independent
- Accounts for popular entities (e.g., Justin Bieber)



## Issue 2: Differentiating between events

- How to differentiate between two events that **cover the same relationships?**
- **Example:** Robert and Susan Downey
  - **Event 1:** Wedding, August 27, 2005
  - **Event 2:** Minor charity event, Nov 11, 2006
- We need to be able to distinguish these!

# Scoring of event timestamps

- Further improvement when we not only score relationships but also **score the event timestamp**

$$F(S) = \sum_{r \in R} w_R(r) + \sum_{e \in S} w_T(t_e)$$

where

$$R = \bigcup_{e \in S} X_e$$



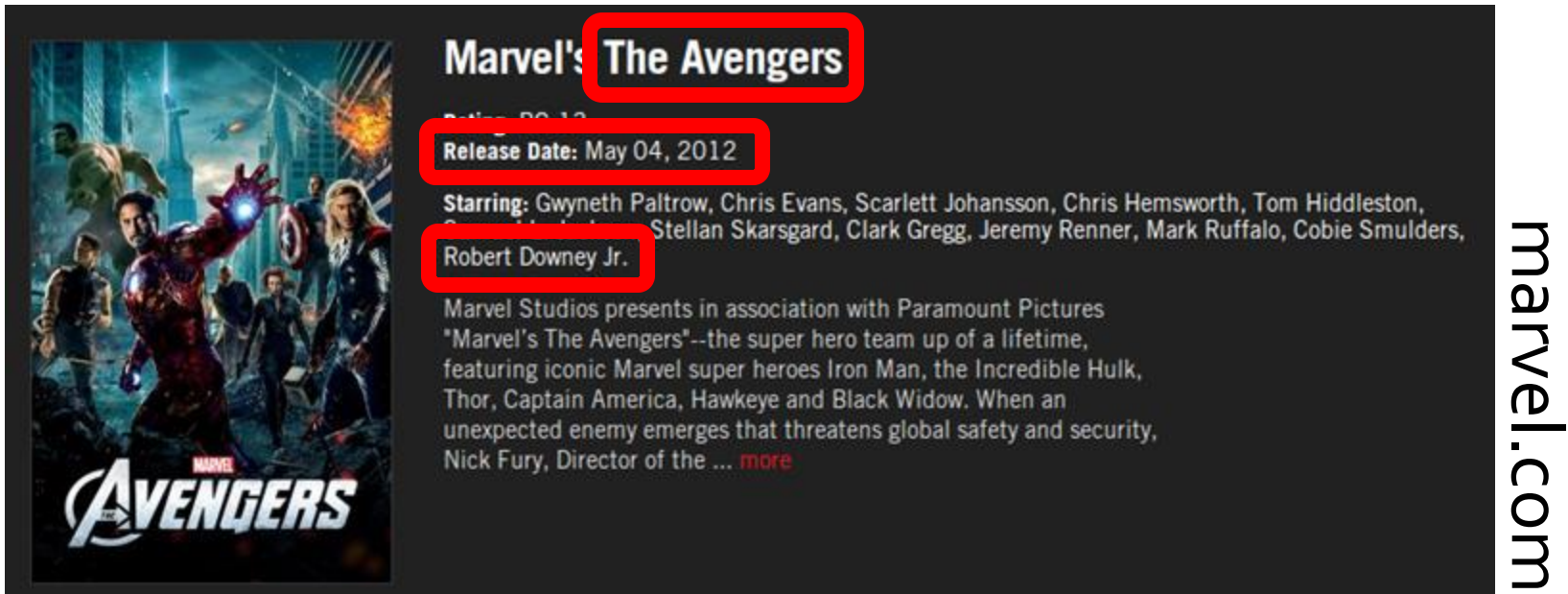
Relationship (as before)



Timestamps

- Again, use co-occurrences for weights  $w_T$

# Co-occurrences on Web Scale



Marvel's **The Avengers**

Release Date: May 04, 2012

Starring: Gwyneth Paltrow, Chris Evans, Scarlett Johansson, Chris Hemsworth, Tom Hiddleston, Samuel L. Jackson, Stellan Skarsgard, Clark Gregg, Jeremy Renner, Mark Ruffalo, Cobie Smulders, **Robert Downey Jr.**

Marvel Studios presents in association with Paramount Pictures "Marvel's The Avengers"--the super hero team up of a lifetime, featuring iconic Marvel super heroes Iron Man, the Incredible Hulk, Thor, Captain America, Hawkeye and Black Widow. When an unexpected enemy emerges that threatens global safety and security, Nick Fury, Director of the ... [more](#)

marvel.com

- “Robert Downey Jr” and “May 4, 2012” occurs 173 times on 71 different webpages
- US Release date of *The Avengers*
- Use MapReduce on 10B web pages (10k+ machines)

# Complete Optimization Problem

- Generalized earlier **coverage** function to **linear combination** of **weighted coverage** functions

$$F(S) = \sum_{r \in R} w_R(r) + \sum_{e \in S} w_T(t_e) \quad \text{where} \quad R = \bigcup_{e \in S} X_e$$

- **Goal:**  $\max_{|S| \leq k} F(S)$
- **Still NP-hard**  
(because generalization of NP-hard problem)

# Next

- How can we **actually optimize** this function?
- What **structure** is there that will help us do this efficiently?
  
- **Any questions so far?**

# Approximate Solution

- For this optimization problem, Greedy produces a solution  $S$   
s.t.  $F(S) \geq (1-1/e)*OPT$  ( $F(S) \geq 0.63*OPT$ )  
[Nemhauser, Fisher, Wolsey '78]
- Claim holds for functions  $F(\cdot)$  which are:
  - *Submodular, Monotone, Normal, Non-negative*  
(discussed next)

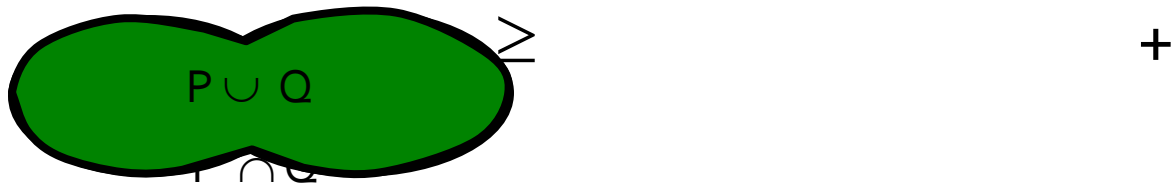
# Submodularity: Definition 1

## Definition:

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

For all  $P, Q \subseteq U$ :

$$F(P) + F(Q) \geq F(P \cup Q) + F(P \cap Q)$$



# Submodularity: Definition 2

- Checking the previous definition is not easy in practice
- Substitute  $P = A \cup \{d\}$  and  $Q = B$  where  $A \subseteq B$  and  $d \notin B$  in the definition above

From before:  $F(P) + F(Q) \geq F(P \cup Q) + F(P \cap Q)$

$$F(A \cup \{d\}) + F(B) \geq F(A \cup \{d\} \cup B) + F((A \cup \{d\}) \cap B)$$

$$F(A \cup \{d\}) + F(B) \geq F(B \cup \{d\}) + F(A)$$

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

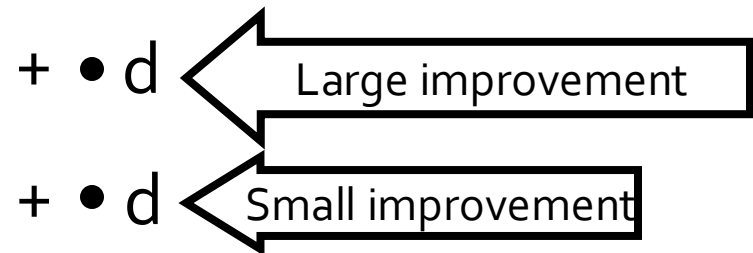
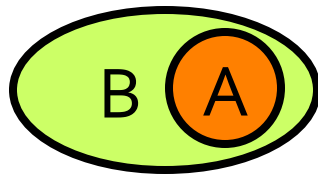
Common definition of Submodularity



# Submodularity: Definition 2

## ■ Diminishing returns characterization

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

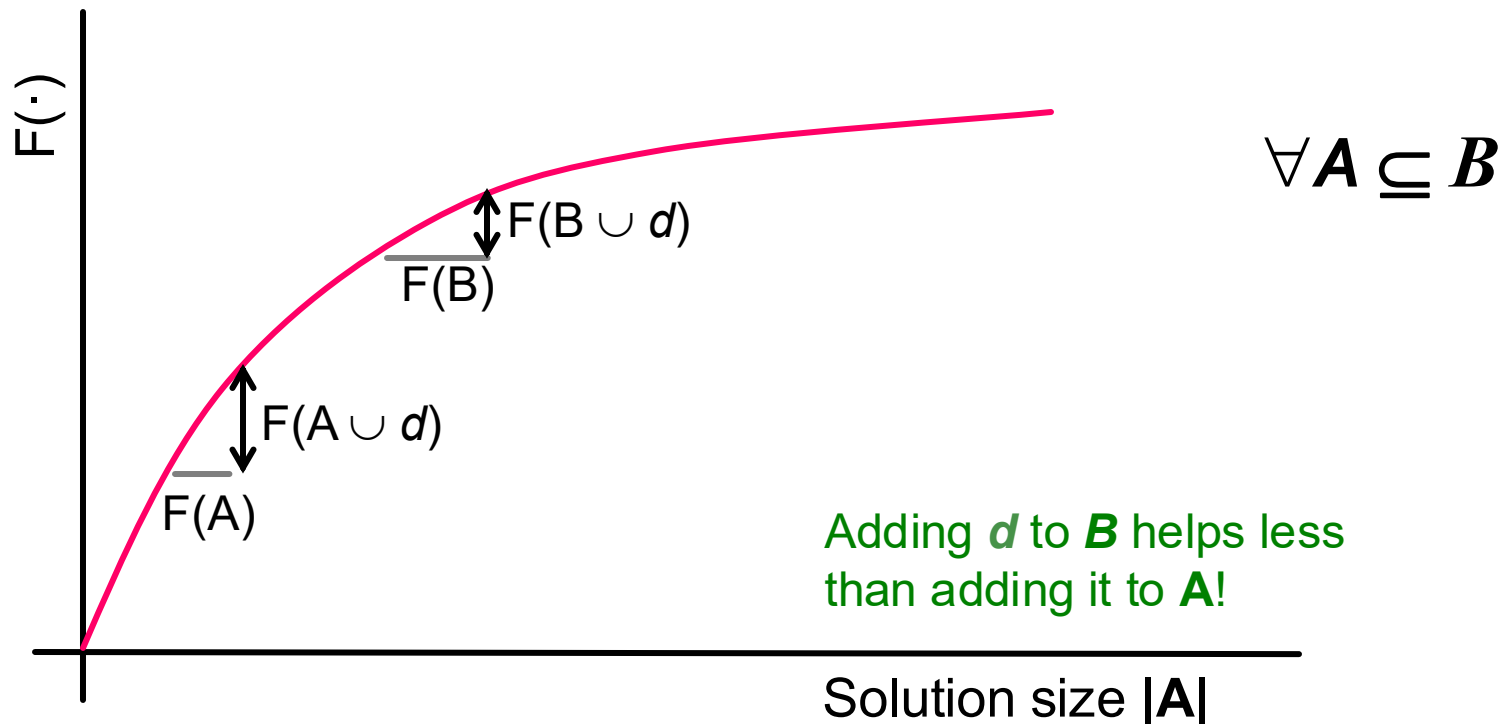


# Submodularity: Diminishing Returns

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

Gain of adding  $d$  to a small set

Gain of adding  $d$  to a large set



# Submodularity: An important property

Let  $F_1 \dots F_M$  be **submodular functions** and  $\lambda_1 \dots \lambda_M \geq 0$  and let  $S$  denote some solution set, then the non-negative linear combination  $F(S)$  (defined below) of these functions is also **submodular**.

$$F(S) = \sum_{i=1}^M \lambda_i F_i(S)$$

# Submodularity: Approximation Guarantee

- When maximizing a submodular function with cardinality constraints, Greedy produces a solution  $S$  for which  $F(S) \geq (1-1/e)*OPT$  i.e.,  $(F(S) \geq 0.63*OPT)$

[Nemhauser, Fisher, Wolsey '78]

- **Claim holds for functions  $F(\cdot)$  which are:**
  - **Monotone:** if  $A \subseteq B$  then  $F(A) \leq F(B)$
  - **Normal:**  $F(\{\})=0$
  - **Non-negative:** For any  $A$ ,  $F(A) \geq 0$
  - **In addition to being submodular**

**Back to our Timeline Problem**

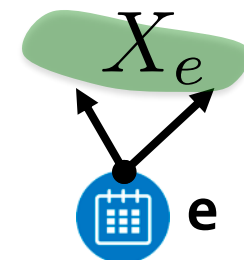
---

# Simple Coverage Model

- Suppose we are given a set of events  $E$ 
  - Each event  $e$  covers a set  $X_e$  of relationships  $\mathbf{U}$
- For a set of events  $S \subseteq E$  we define:

$$F(S) = \left| \bigcup_{e \in S} X_e \right|$$

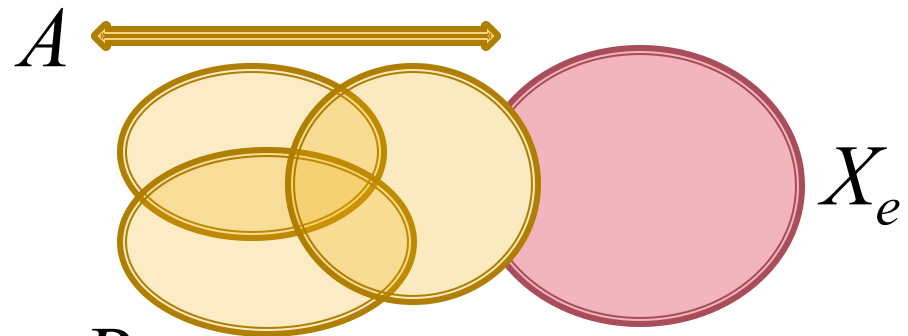
- Goal: We want to  $\max_{|S| \leq k} F(S)$  ← Cardinality Constraint
- Note:  $F(S)$  is a set function:  $F(S) : 2^E \rightarrow \mathbb{N}$



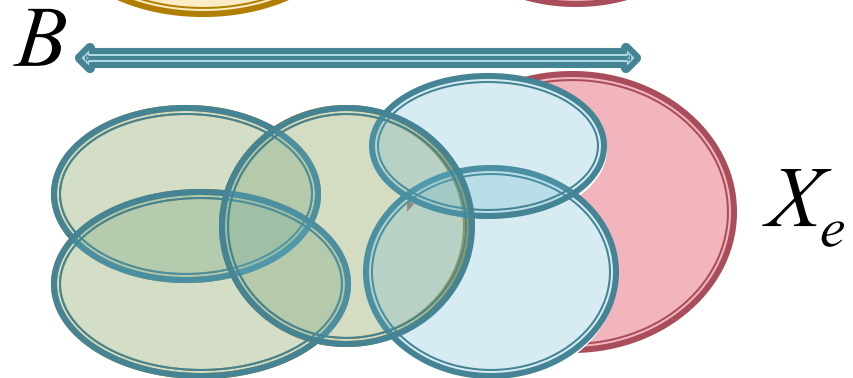
# Simple Coverage: Submodular?

- **Claim:**  $F(S) = \left| \bigcup_{e \in S} X_e \right|$  is submodular.

*Gain of adding  $X_e$  to a smaller set*



*Gain of adding  $X_e$  to a larger set*



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

$$\forall A \subseteq B$$

# Simple Coverage: Other Properties

- **Claim:**  $F(S) = \left| \bigcup_{e \in S} X_e \right|$  is normal & monotone
- **Normality:** When  $S$  is empty,  $\bigcup_{e \in S} X_e$  is empty.
- **Monotonicity:** Adding a new event to  $S$  can never decrease the number of relationships covered by  $S$ .
- **What about non-negativity?**

**Monotone:** if  $A \subseteq B$  then  $F(A) \leq F(B)$

**Normal:**  $F(\{\}) = 0$

**Non-negative:** For any  $A$ ,  $F(A) \geq 0$



# Summary so far

|               | Simple Coverage | Weighted Coverage (Relationships) | Weighted Coverage (Timestamps) | Complete Optimization Problem |
|---------------|-----------------|-----------------------------------|--------------------------------|-------------------------------|
| Submodularity | ✓               |                                   |                                |                               |
| Monotonicity  | ✓               |                                   |                                |                               |
| Normality     | ✓               |                                   |                                |                               |

# Weighted Coverage (Relationships)

$$F(S) = \sum_{r \in R} w(r) \quad w : R \rightarrow \mathbb{R}^+ \quad \text{where} \quad R = \bigcup_{e \in S} X_e$$

## ■ Claim: $F(S)$ is submodular.

- Consider two sets  $A$  and  $B$  s.t.  $A \subseteq B \subseteq S$  and let us consider an event  $e \notin B$
- Three possibilities when we add  $e$  to  $A$  or  $B$ :
  - **Case 1:**  $e$  does not cover any new relationships w.r.t both  $A$  and  $B$   
 $F(A \cup \{e\}) - F(A) = 0 = F(B \cup \{e\}) - F(B)$

# Weighted Coverage (Relationships)

$$F(S) = \sum_{r \in R} w(r) \quad w : R \rightarrow \mathbb{R}^+$$

- **Claim:  $F(S)$  is submodular.**

- Three possibilities when we add  $e$  to  $A$  or  $B$ :
  - **Case 2:**  $e$  covers some new relationships w.r.t  $A$  but not w.r.t  $B$

$$F(A \cup \{e\}) - F(A) = \nu \text{ where } \nu \geq 0$$

$$F(B \cup \{e\}) - F(B) = 0$$

$$\text{Therefore, } F(A \cup \{e\}) - F(A) \geq F(B \cup \{e\}) - F(B)$$

# Weighted Coverage (Relationships)

$$F(S) = \sum_{r \in R} w(r) \quad w : R \rightarrow \mathbb{R}^+$$

- **Claim:  $F(S)$  is submodular.**

- Three possibilities when we add  $e$  to  $A$  or  $B$ :
  - **Case 3:**  $e$  covers some new relationships w.r.t both  $A$  and  $B$

$$F(A \cup \{e\}) - F(A) = v \text{ where } v \geq 0$$

$$F(B \cup \{e\}) - F(B) = u \text{ where } u \geq 0$$

But,  $v \geq u$  because  $e$  will always cover fewer new relationships w.r.t  $B$  than w.r.t  $A$  because  $A \subseteq B$

# Weighted Coverage (Relationships)

$$F(S) = \sum_{r \in R} w(r) \quad w : R \rightarrow \mathbb{R}^+ \quad \text{where} \quad R = \bigcup_{e \in S} X_e$$

- **Claim:**  $F(S)$  is monotone and normal.
- **Normality:** When  $S$  is empty,  $R = \bigcup_{e \in S} X_e$  is empty.
- **Monotonicity:** Adding a new event to  $S$  can never decrease the number of relationships covered by  $S$ .

# Summary so far

|               | Simple Coverage | Weighted Coverage (Relationships) | Weighted Coverage (Timestamps) | Complete Optimization Problem |
|---------------|-----------------|-----------------------------------|--------------------------------|-------------------------------|
| Submodularity | ✓               | ✓                                 |                                |                               |
| Monotonicity  | ✓               | ✓                                 |                                |                               |
| Normality     | ✓               | ✓                                 |                                |                               |

# Weighted Coverage (Timestamps)

$$F(S) = \sum_{e \in S} w_T(t_e)$$

- **Claim:  $F(S)$  is submodular, monotone and normal**
- Analogous arguments to that of weighted coverage (relationships) are applicable

# Summary so far

|               | Simple Coverage | Weighted Coverage (Relationships) | Weighted Coverage (Timestamps) | Complete Optimization Problem |
|---------------|-----------------|-----------------------------------|--------------------------------|-------------------------------|
| Submodularity | ✓               | ✓                                 | ✓                              |                               |
| Monotonicity  | ✓               | ✓                                 | ✓                              |                               |
| Normality     | ✓               | ✓                                 | ✓                              |                               |



# Complete Optimization Problem

- Generalized earlier **coverage** function to non-negative **linear combination** of **weighted coverage** functions

$$F(S) = \boxed{F_1(S)} + \boxed{F_2(S)} \quad \text{where} \quad R = \bigcup_{e \in S} X_e$$

- Goal:**  $\max_{|S| \leq k} F(S)$
- Claim:**  $F(A)$  is submodular, monotone and normal

# Complete Optimization Problem

- **Submodularity:**  $F(S)$  is a non-negative linear combination of two submodular functions. Therefore, it is submodular too.
- **Normality:**  $F_1(\{\}) = 0 = F_2(\{\})$   
 $F_1(\{\}) + F_2(\{\}) = 0$
- **Monotonicity:** Let  $A \subseteq B \subseteq S$ ,  
 $F_1(A) \leq F_1(B)$  and  $F_2(A) \leq F_2(B)$   
 $F_1(A) + F_2(A) \leq F_1(B) + F_2(B)$

# Summary so far

|               | Simple Coverage | Weighted Coverage (Relationships) | Weighted Coverage (Timestamps) | Complete Optimization Problem |
|---------------|-----------------|-----------------------------------|--------------------------------|-------------------------------|
| Submodularity | ✓               | ✓                                 | ✓                              | ✓                             |
| Monotonicity  | ✓               | ✓                                 | ✓                              | ✓                             |
| Normality     | ✓               | ✓                                 | ✓                              | ✓                             |

# Lazy Optimization of Submodular Functions

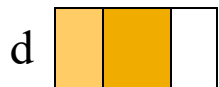
---

# Greedy Solution

## Greedy

Marginal gain:

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



Add element with  
highest marginal gain

- *Greedy Algorithm is Slow!*
- At each iteration, we need to evaluate marginal gains of all the remaining elements
- Runtime  $O(|U| * K)$  for selecting  $K$  elements out of the set  $U$

# Speeding up Greedy

- ***In round  $i$ :***
  - So far we have  $S_{i-1} = \{e_1 \dots e_{i-1}\}$
  - Now we pick an element  $e \notin S_{i-1}$  which maximizes the marginal benefit  $\Delta_i = F(S_{i-1} \cup \{e\}) - F(S_{i-1})$
- ***Key observation:***
  - ***Marginal gain of any element  $e$  can never increase!***
  - For every element  $e$ :  
 $\Delta_i(e) \geq \Delta_j(e)$  for all iterations  $i < j$

# Lazy Greedy

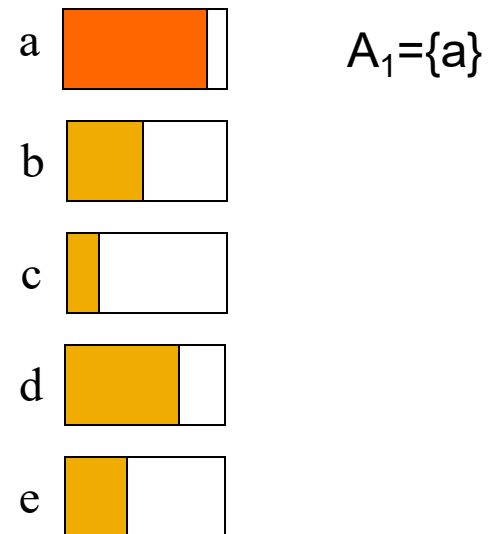
## ■ Idea:

- 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

Upper bound on  
Marginal gain  $\Delta_1$



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

# Lazy Greedy

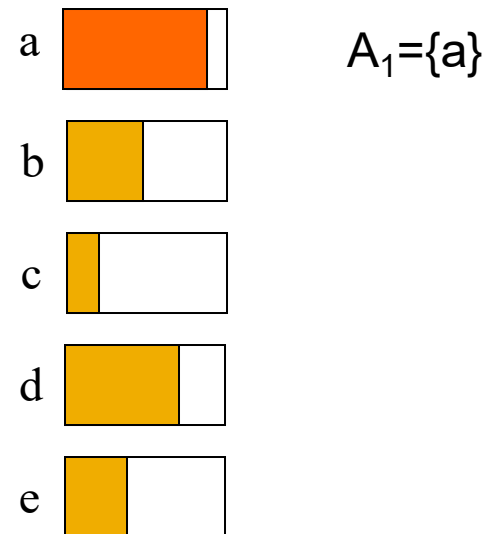
## ■ Idea:

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

## ■ Lazy Greedy:

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

Upper bound on  
Marginal gain  $\Delta_2$



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



# Lazy Greedy

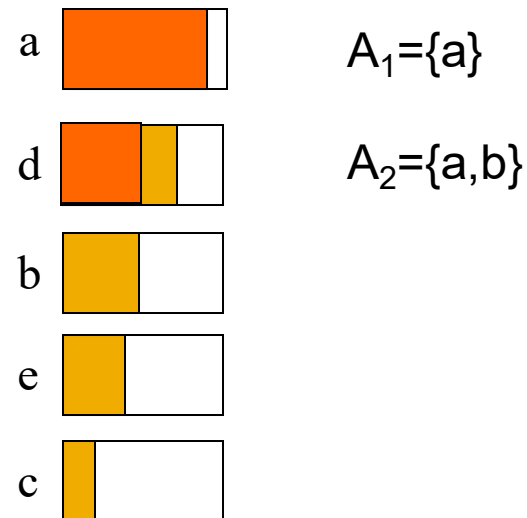
## ■ Idea:

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

## ■ Lazy Greedy:

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

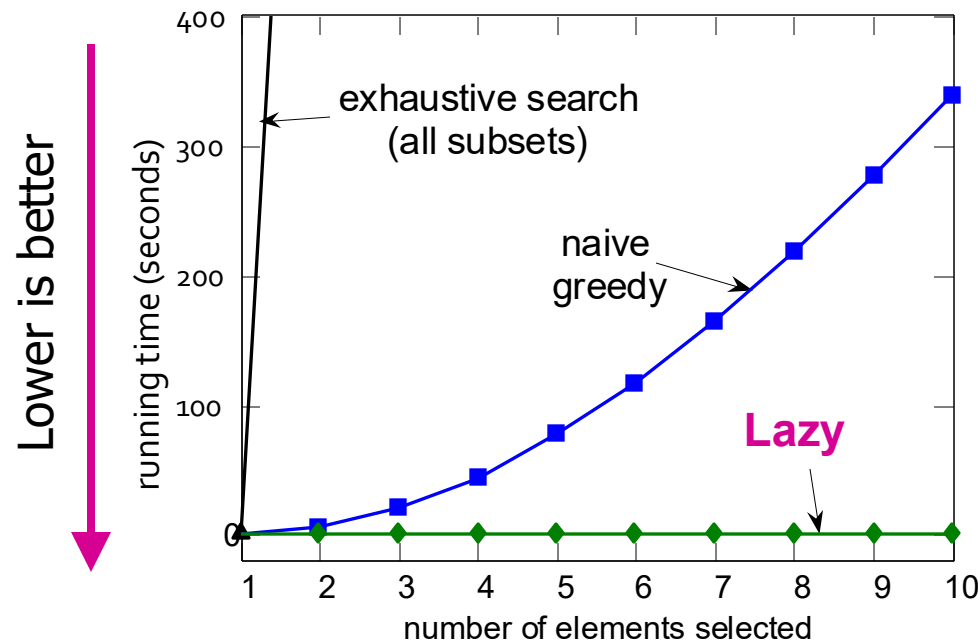
Upper bound on  
Marginal gain  $\Delta_2$



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

# Speed Up of Lazy Greedy Algorithm

- Lazy greedy offers significant speed-up over traditional greedy implementations in practice.



# References

- Althoff et. al., TimeMachine: Timeline Generation for Knowledge-Base Entities, KDD 2015
- Leskovec et. al., Cost-effective Outbreak Detection in Networks, KDD 2007
- Andreas Krause, Daniel Golovin, Submodular Function Maximization
- ICML Tutorial:  
<http://submodularity.org/submodularity-icml-part1-slides-prelim.pdf>
- Learning and Testing Submodular Functions:  
<http://grigory.us/cis625/lecture3.pdf>
- UW Research by Jeff Bilmes (ECE)