Influence maximization

Mario Michelessa

Context

- Influence maximization :

  Given G = (V,E,P), $k \geq 1$, find $\text{argmax}_{S \subset V, |S|=k}\ \sigma(S)$ with σ being the influence function.

- $P = (p_{u,v})$ = diffusion probability = probability of u influencing v when u posts something
- $\sigma(S) = \sum_{v \in V}\left(1 - \prod_{u \in S}\left(1 - p_{u,v}\right)\right)$ under the assumption that the graph is bipartite [1]

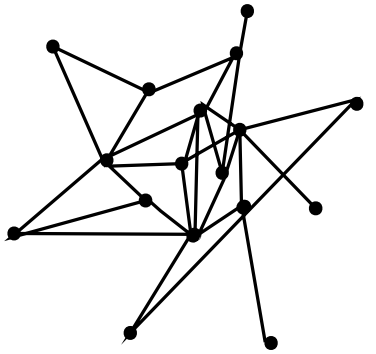
Sakaue, Differentiable Greedy Submodular Maximization: Guarantees, Gradient Estimators, and Applications, link

Context

Goal:

Given $G_s = (V, E_s)$ the social graph, and (C_i) list of cascades,

Find model $m(.,w)$ st $m(features(u,v),w) = p_{u,v}$



social graph

First step :

Find features

Subsampling of graph + labels

User profile information

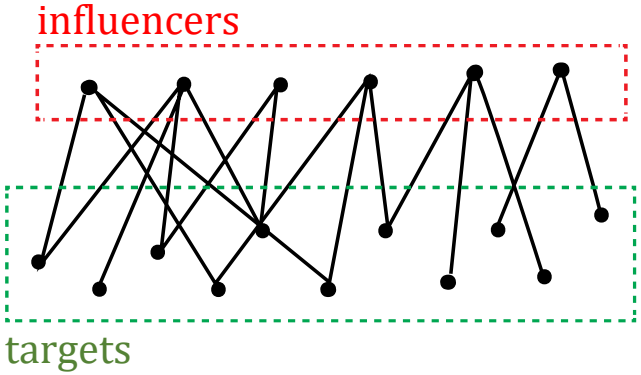
Information about topic

Social graph information

Train the model

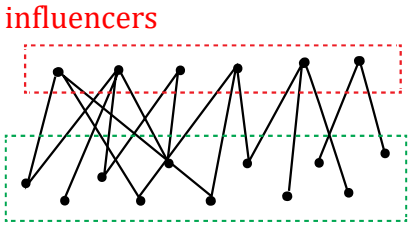
2-stage framework

decision-focused framework



graph induced by (C_i)

Preprocessing



graph induced by (C_i)

Cascade information

We call **influencer** a user initiating a cascade and **target** a user participating in the cascade.

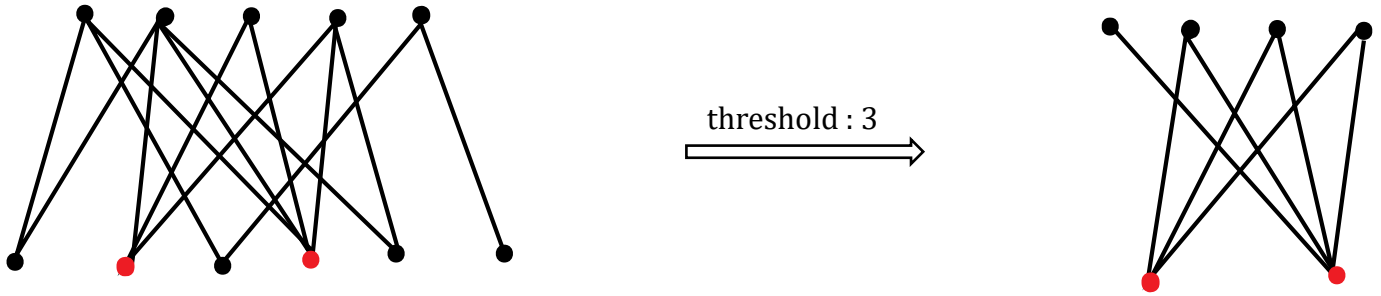
=> total dataset : 300K cascades, 44K different influencers and 1.7M targets.

We only keep a small subset of users : only the targets participating in **more than 200 cascades** are selected.

=> 8K influencers and 1800 targets

Ultimately, the model will not use the cascades information, they are only used for deducing the ground truth diffusion probabilities

Example :



Dataset :

#influencers = 44K

#targets 1.7M

threshold : 200

#influencers = 8K

#targets = 1800

Preprocessing

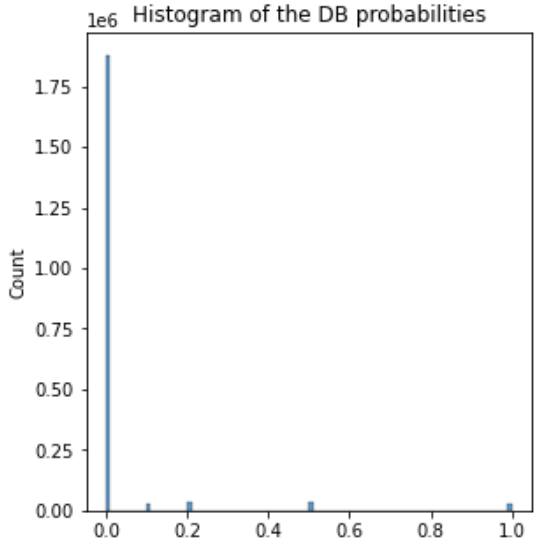
The model needs ground-truth diffusion probabilities, 2 methods have been implemented to estimate those :

- Data-based method

$$p_{u,v} = \frac{\#times\ v\ reposted\ u's\ posts}{\#times\ u\ posted + \#times\ v\ reposted}$$

However, the probabilities are very low due to the high number of actions made by u and v.
In order to counter that, we box the values into 5 categories.

If p = 0, it stays at 0
If p > 0 :
If p < low_quantile then p = 0.1
ElseIf p < med_quantile then p = low_prob
ElseIf p < high_quantile then p = med_prob
Else p = high_prob



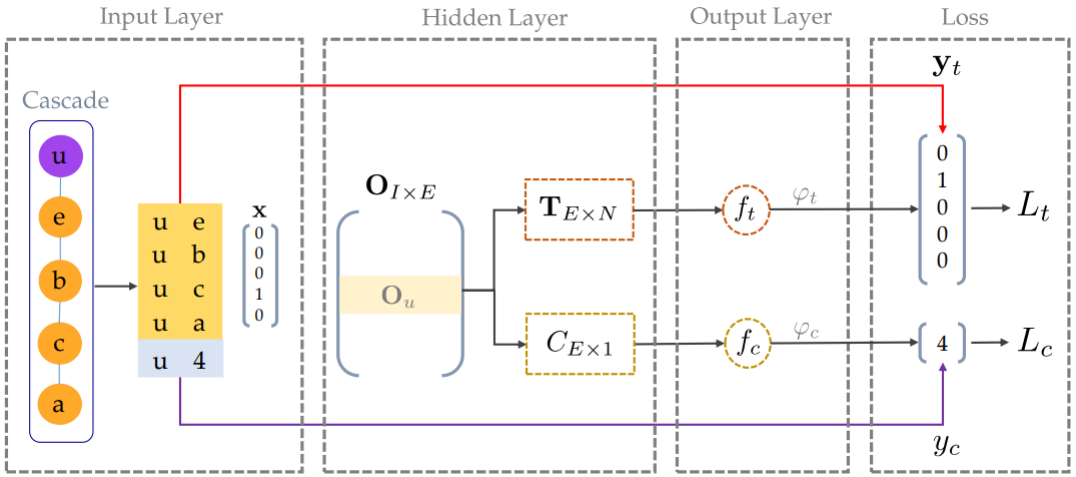
Goyal, Bonchi, Lakshmanan - Learning Influence Probabilities In Social Networks, link
Ko, Lee, Shin, Park - MONSTOR: An Inductive Approach for Estimating and Maximizing Influence over Unseen Networks, link

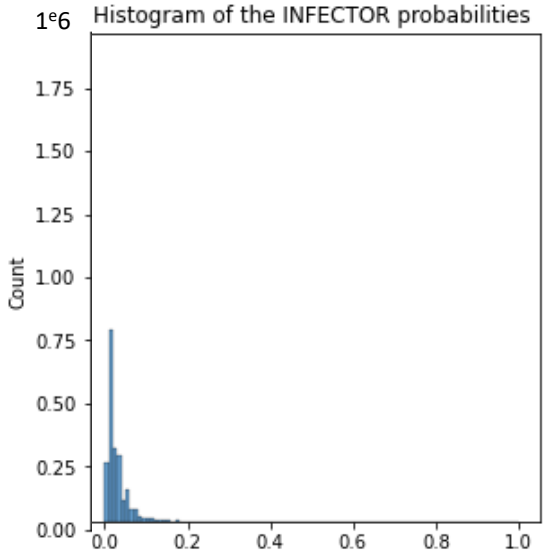
Preprocessing

The model needs ground-truth diffusion probabilities, 2 methods have been implemented to estimate those :

- INFECTOR method

$$p_{u,v} = \frac{\exp(E_u^T E_v)}{\sum_{v \in T} \exp(E_u^T E_v)}$$ with E_u being the embedding vector of u estimated by the INFECTOR model





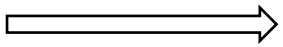
Panagopoulos, Malliaros, Vazirgiannis - Multi-task Learning for Influence Estimation and Maximization, link

Preprocessing

- User profile information

A file contains information about the user profiles :

uid	int64
bi_followers_count	int32
city	category
verified	category
followers_count	int32
location	object
province	category
friends_count	int32
name	object
gender	category
created_at	object
verified_type	category
statuses_count	int32
description	object



From these information, we create 5 features :

$\log(followersCount + 1)$
$\log(friendsCount + 1)$
$\log(statusesCount + 1)$
$verified \in \{0,1\}$
$gender \in 0,1$

Preprocessing

Social graph information : “who follows whom?”

Given this induced graph on our subsampled dataset, we estimate 3 other features :

For each selected influencer, we estimate

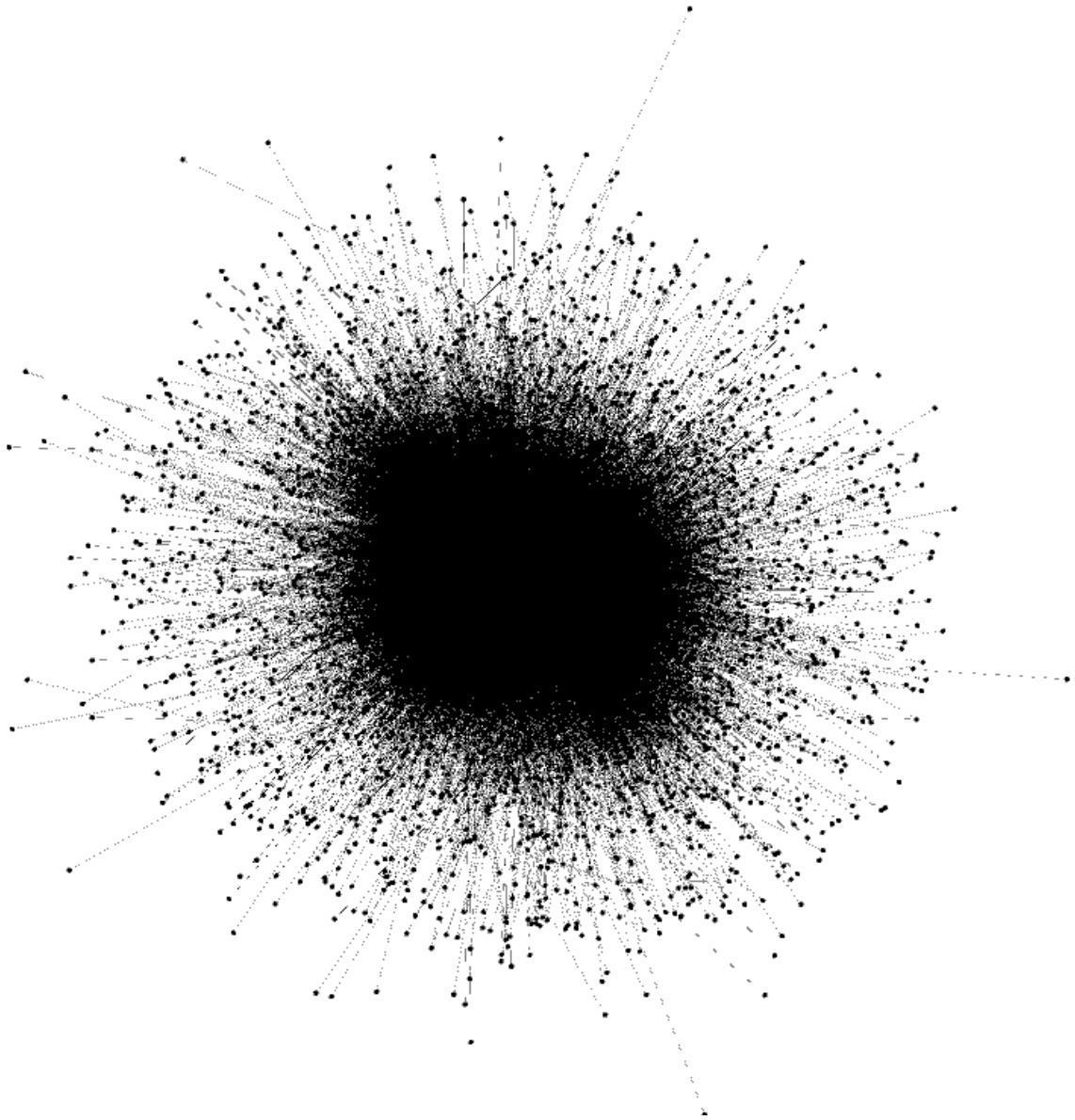
ingoing degree
pagerank
number of nodes at distance less than 2

For each selected target, we estimate

outgoing degree
pagerank

For each pair of (influencer, target) we store

1 if an edge exists in the social graph, or 0 if not.



ZHOU, XU, TRAJCEVSKI, ZHANG - A Survey of Information Cascade Analysis: Models, Predictions, and Recent Advances, link

Preprocessing

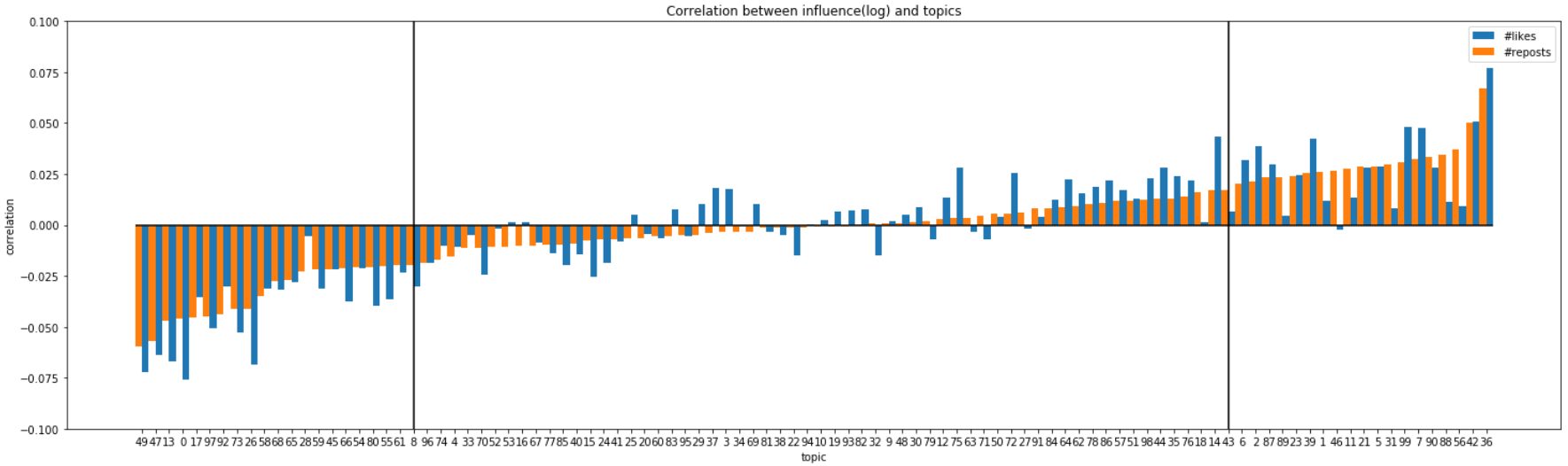
Topic information

Input :

topic distribution for each message = 100-vector where the i-th coefficient is the probability the message is in the i-th category.

- Step 1 : The 100-vectors are transformed in a 3-vector
- Step 2 : Creation of users features with these messages' features

Step 1 : The 100-vectors are transformed in a 3-vector



- **Highly influencial topics (HT)**: The top 20 categories most positively correlated with influence (estimated by the number of targets/cascades and number of likes)
- **Low influencial topics (LT)**: The top 20 categories most negatively correlated with influence
- **Neutral topics (NT)**: The remaining 60 topics that are uncorrelated with influence

For each message, the new topic distribution is estimated by summing the probabilities of the topics in the same category.

$$t_m{}' = (p'_{HT}, p'_{NT}, p'_{LT}) = \left(\sum_{t \in HT} p_t \, , \sum_{t \in NT} p_t \, , \sum_{t \in LT} p_t \right)$$

Step 2 : : Creation of users features from the message's features

message's topic classification : $(x_{HT}, x_{NT}, x_{LT}) \in [0,1]^3$

- For influencers, we estimate the average of all the topic distribution from the emitted message

  influencer's u topic classification $= mean(t'_m \; for \; m \; sent \; by \; u)$

- And for targets, we estimate the average of all the topic distribution from the messages they reacted to

  target's v topic classification $= mean(t'_m \; for \; m \; reposted \; by \; v)$

Feature vector

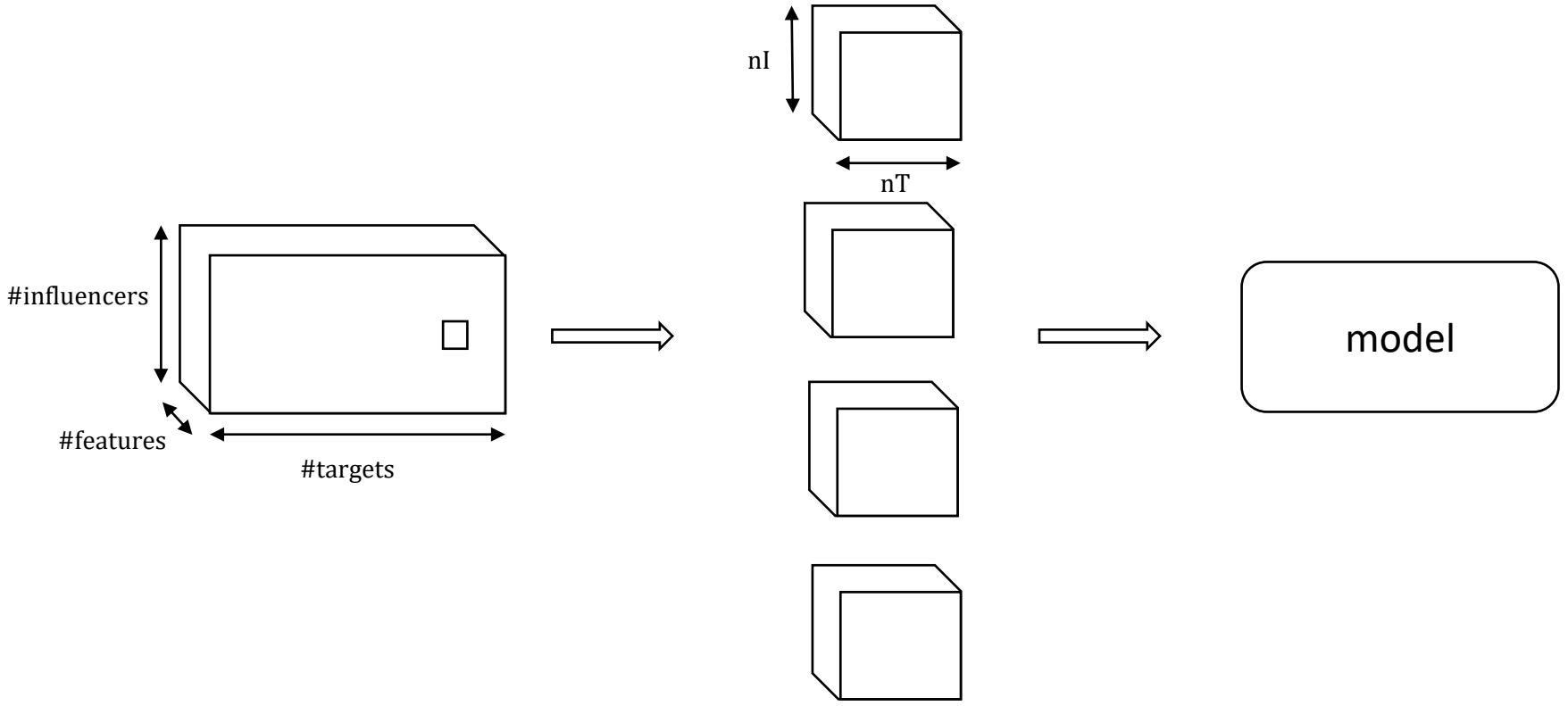
For each pair of influencer/target (u,v) the feature vector is then the concatenation of u's features, v's features and the pair (u,v)'s feature

$$features(u, v)$$

u	followers_count
u	friends_count
u	statuses_count
u	verified
u	gender
u	outgoing degree
u	pagerank
u	# 2-reachable nodes
u	high inf topic
u	neutral topic
u	low inf topic
v	followers_count
v	friends_count
v	statuses_count
v	verified
v	gender
v	ingoing degree
v	pagerank
v	high inf topic
v	neutral topic
v	low inf topic
u,v	1(v follows u)

Training of the model

- The model is fed with multiple instances of fixed sizes, representing random subsampling of the graph
  - nT targets are randomly chosen among the targets
  - nI influencers are randomly chosen among the 20% influencers having the most numerous targets in their cascades
  - For the rest of the presentation, $nI = nT = 500$



Sakaue, Differentiable Greedy Submodular Maximization: Guarantees, Gradient Estimators, and Applications, link

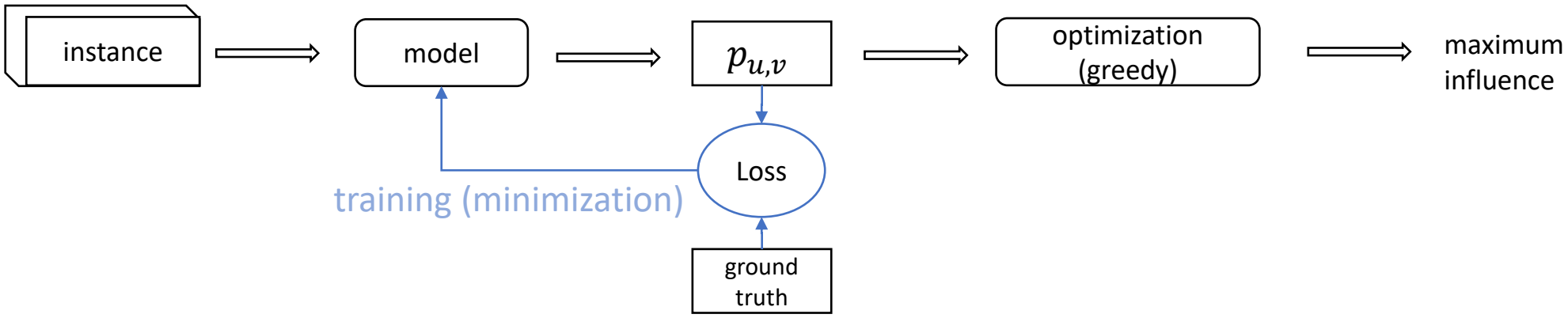
Loss function

2 types of training frameworks are studied :

- 2-stage models
- Decision focused models

They differ by their loss function.

2-stage Model : Maximizes accuracy of model and then uses the output in an optimization algorithm to maximize influence



Loss function : MSE

Or unbalanced loss : $L(p_{pred}, p_{th}) = \exp\left(\phi * (p_{th} - p_{pred})\right) - \phi * (p_{th} - p_{pred}) - 1$

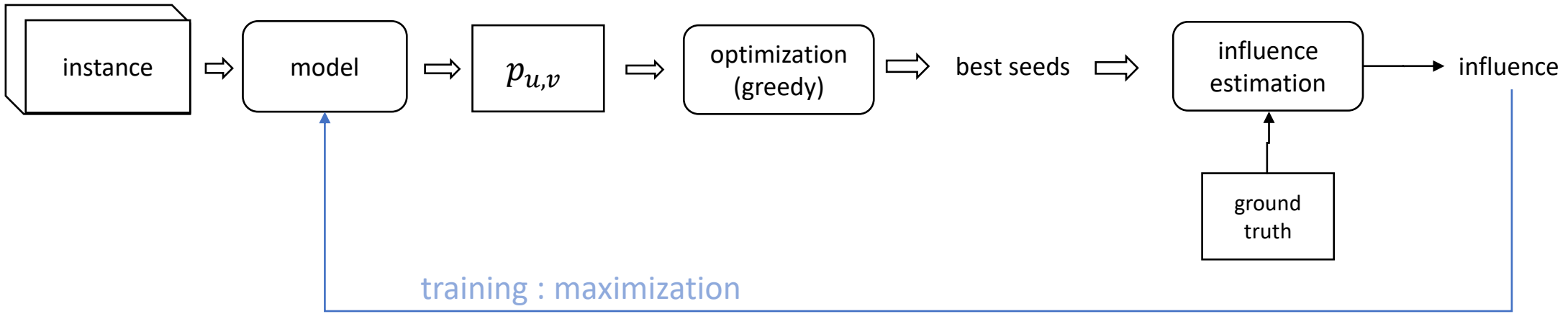
Loss function

2 types of training frameworks are studied :

- 2-stage models
- Decision focused models

They differ by their loss function.

decision-focused model: returns the diffusion probability matrix giving the best performing seeds on the real network



Gradient : $$-\nabla_{\mathbf{w}}\mathbb{E}_{S\sim p(m(\mathbf{X}_i,\mathbf{w}))}[f(S,\boldsymbol{\theta}_i)] \approx -\frac{1}{N}\sum_{j=1}^{N} f(S_j,\boldsymbol{\theta}_i)\nabla_{\boldsymbol{\theta}} \ln p(S_j,\boldsymbol{\theta})|_{\boldsymbol{\theta}=m(\mathbf{X}_i,\mathbf{w})} \cdot \nabla_{\mathbf{w}} m(\mathbf{X}_i,\mathbf{w}).$$

Sakaue, Differentiable Greedy Submodular Maximization: Guarantees, Gradient Estimators, and Applications, link

Metrics

Quality metrics: measure the quality of seeds chosen.

2 methods :

Estimating the expectation of influenced nodes given that the graph is bipartite

$$\sigma(S) = \sum_{v \in T} \left(1 - \prod_{u \in S} (1 - p_{u,v}) \right)$$

Using the cascade data and computing the number of Different Nodes Influenced (DNI)

For a seed set S,

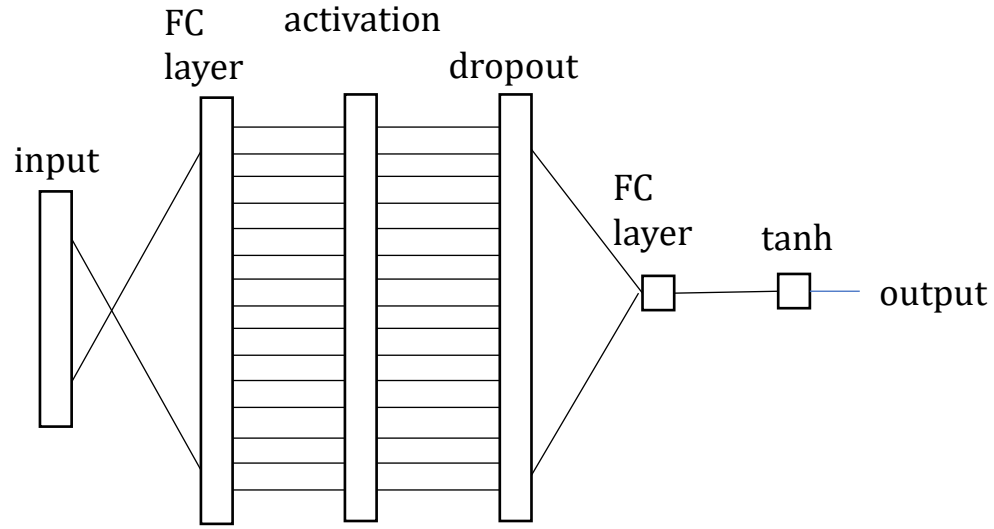
$$DNI(S) = \#targets\ v\ st\ v\ appears\ in\ at\ least\ one\ cascade\ posted\ by\ u \in S$$

Panagopoulos, Malliaros, Vazirgiannis - Multi-task Learning for Influence Estimation and Maximization, link

Architecture and parameters

Architecture parameters :

- dropout proportion
- hidden layer size
- activation function
  - ReLU
  - sigmoid



Optimizer : Adam optimizer

- learning rate
- momentum
- batch size

The differentiable greedy algorithm of the decision-focused loss function has also parameters to tune :

- ϵ = perturbation factor of the distribution.
  It behaves like a temperature. When the temperature decreases, the distribution approaches argmax. When it increases, it approaches a uniform distribution.
- sample_size = number of differentiable greedy algorithms ran at each step.

Results

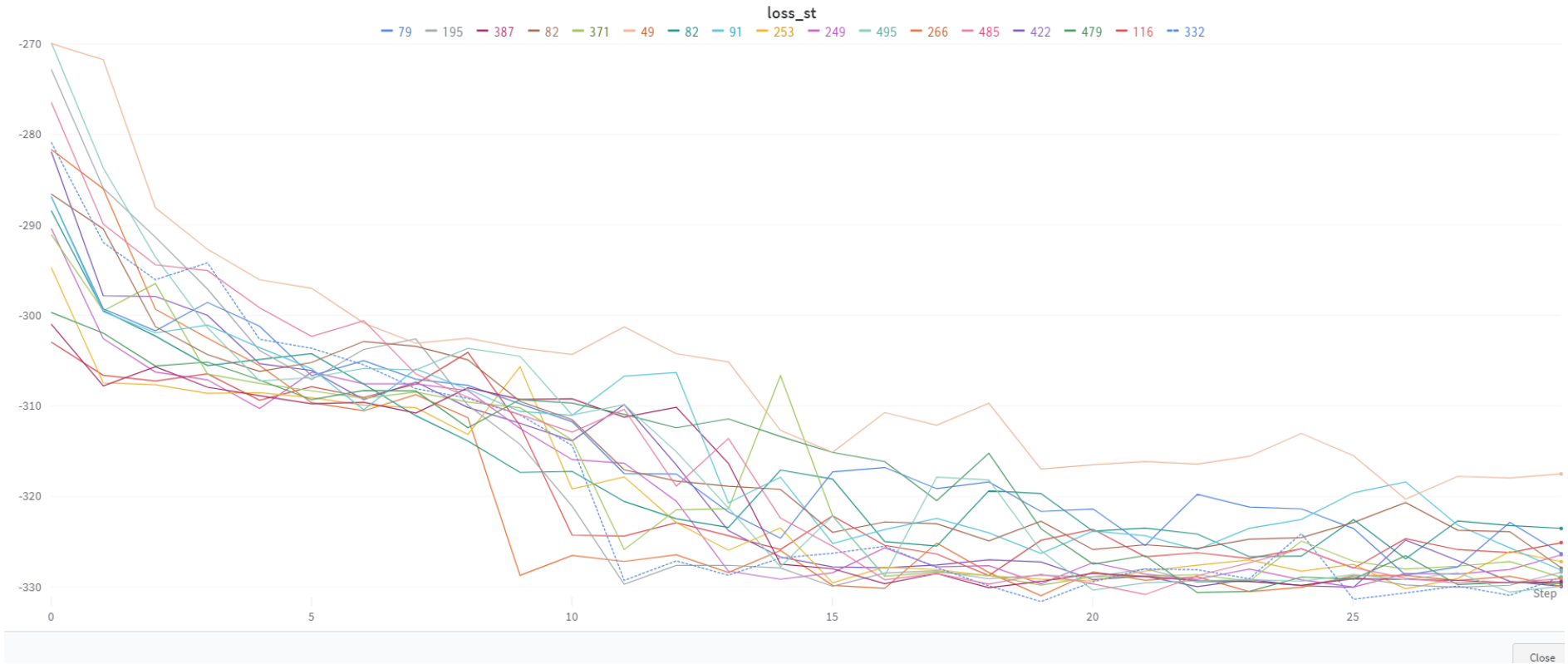
Stochastic greedy algorithm :
Only selects a random subset of candidates among all candidates and then draws the best seed from there.
Supposed to speed up computations : fewer influence estimation executions

Algorithm 1 SMOOTHED GREEDY

$S \leftarrow \emptyset$
for $k = 1, 2 \dots$ **do**
  $U_k = \{u_1, \dots, u_{n_k}\} \leftarrow \{v \notin S \mid S \cup \{v\} \in \mathcal{I}\}$
  $\mathbf{g}_k(\boldsymbol{\theta}) = (g_k(u_1, \boldsymbol{\theta}), \dots, g_k(u_{n_k}, \boldsymbol{\theta})) \leftarrow (f_S(u_1, \boldsymbol{\theta}), \dots, f_S(u_{n_k}, \boldsymbol{\theta}))$
  $\mathbf{p}_k(\boldsymbol{\theta}) = (p_k(u_1, \boldsymbol{\theta}), \dots, p_k(u_{n_k}, \boldsymbol{\theta})) \leftarrow \operatorname{argmax}_{\mathbf{p} \in \Delta^{n_k}} \{\langle \mathbf{g}_k(\boldsymbol{\theta}), \mathbf{p} \rangle - \Omega_k(\mathbf{p})\}$
  $s_k \leftarrow u \in U_k$ with probability $p_k(u, \boldsymbol{\theta})$
  $S \leftarrow S \cup \{s_k\}$
  if S is maximal **then return** S

Algorithm 2 STOCHASTIC SMOOTHED GREEDY

$S \leftarrow \emptyset$
for $k = 1, 2 \dots, K$ **do**
  $U_k = \{u_1, \dots, u_{n_k}\} \leftarrow$ n_k elements chosen from $V \backslash S$ uniformly at random
  $\mathbf{g}_k(\boldsymbol{\theta}) = (g_k(u_1, \boldsymbol{\theta}), \dots, g_k(u_{n_k}, \boldsymbol{\theta})) \leftarrow (f_S(u_1, \boldsymbol{\theta}), \dots, f_S(u_{n_k}, \boldsymbol{\theta}))$
  $\mathbf{p}_k(\boldsymbol{\theta}) = (p_k(u_1, \boldsymbol{\theta}), \dots, p_k(u_{n_k}, \boldsymbol{\theta})) \leftarrow \operatorname{argmax}_{\mathbf{p} \in \Delta^{n_k}} \{\langle \mathbf{g}_k(\boldsymbol{\theta}), \mathbf{p} \rangle - \Omega_k(\mathbf{p})\}$
  $s_k \leftarrow u \in U_k$ with probability $p_k(u, \boldsymbol{\theta})$
  $S \leftarrow S \cup \{s_k\}$
return S

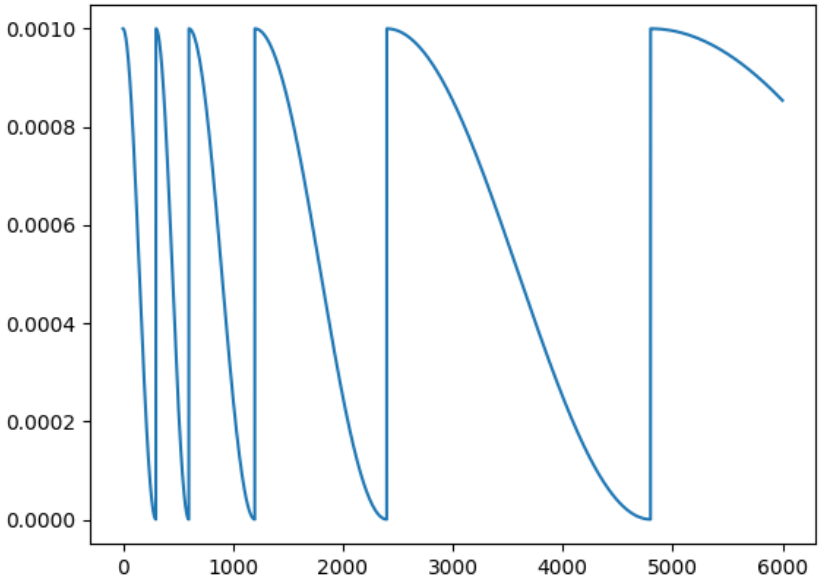


Stochastic greedy algorithm :

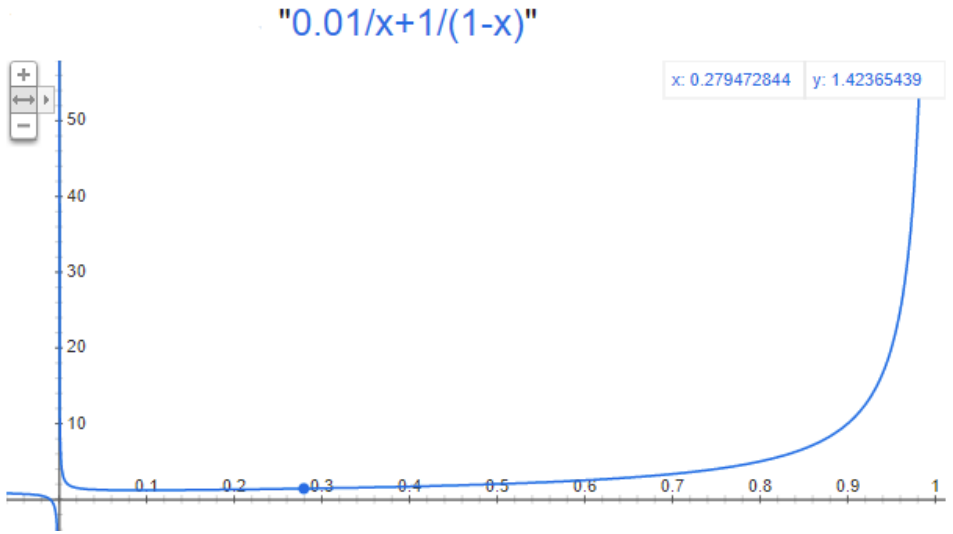
- In practice, the parameter does not deteriorates the training process
- Barely speeds up calculations. Reason : influence function is fast to estimate

Regularization :

Often, the model is stuck in a local minima :
It predicts constant matrices = 1 or 0
1. Cosine annealing with warm restart



2. Penalizing outputs having a mean close to 0 or 1 by adding a term in loss function



Results

Final goal : model that could predict cascades without cascades information

Comparison between models with cascades information and without cascades information :

Adding cascade features :

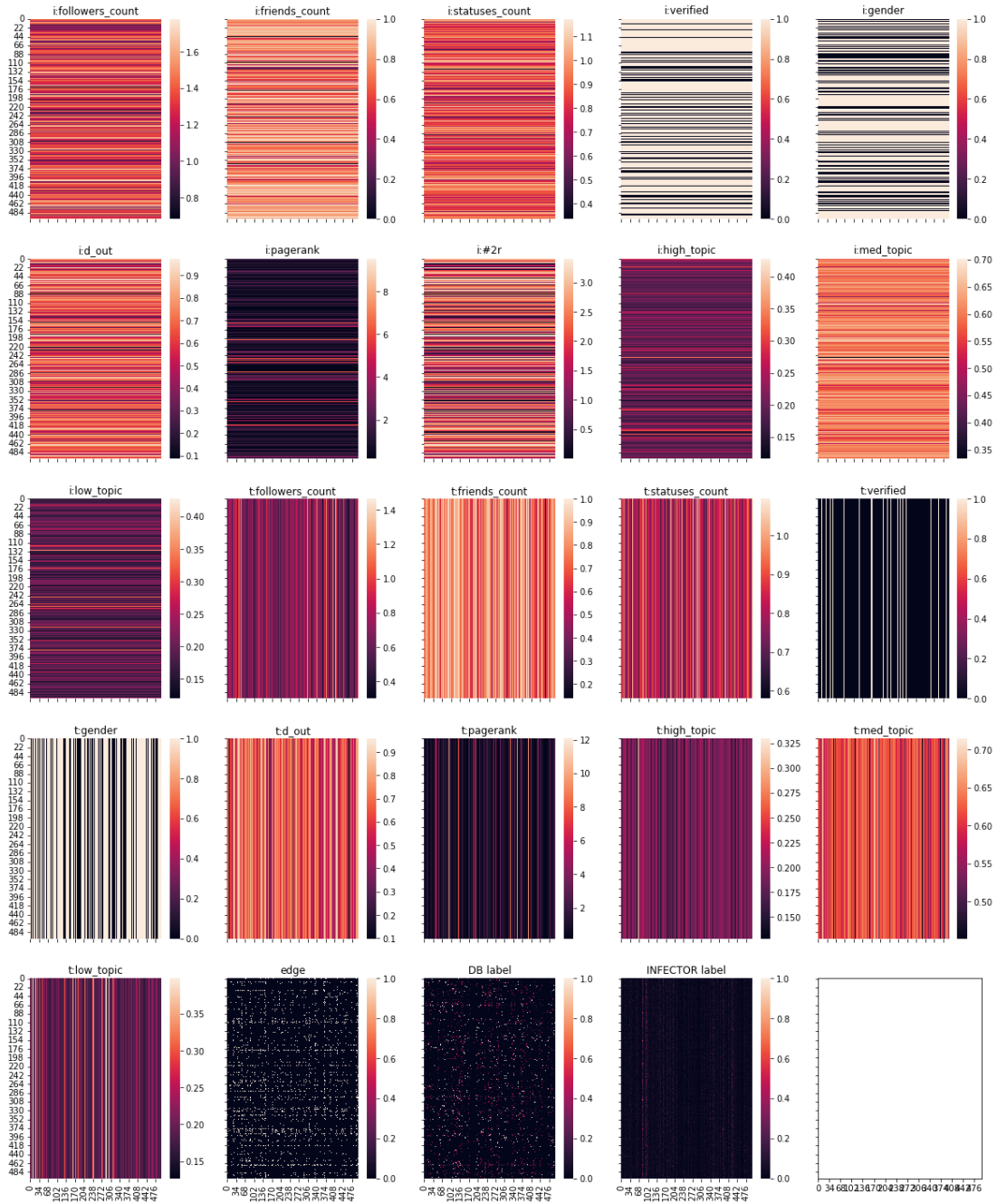
- *nCascades : number of cascades initiated by influencer*
- *totalLikes : sum of likes of cascades initiated by influencer*
- *totalReposts : sum of lengths of cascades initiated by influencer*

Results

Without cascades features

$$features(u, v)$$

u	followers_count
u	friends_count
u	statuses_count
u	verified
u	gender
u	outgoing degree
u	pagerank
u	# 2-reachable nodes
u	high inf topic
u	neutral topic
u	low inf topic
v	followers_count
v	friends_count
v	statuses_count
v	verified
v	gender
v	ingoing degree
v	pagerank
v	high inf topic
v	neutral topic
v	low inf topic
u,v	1(v follows u)

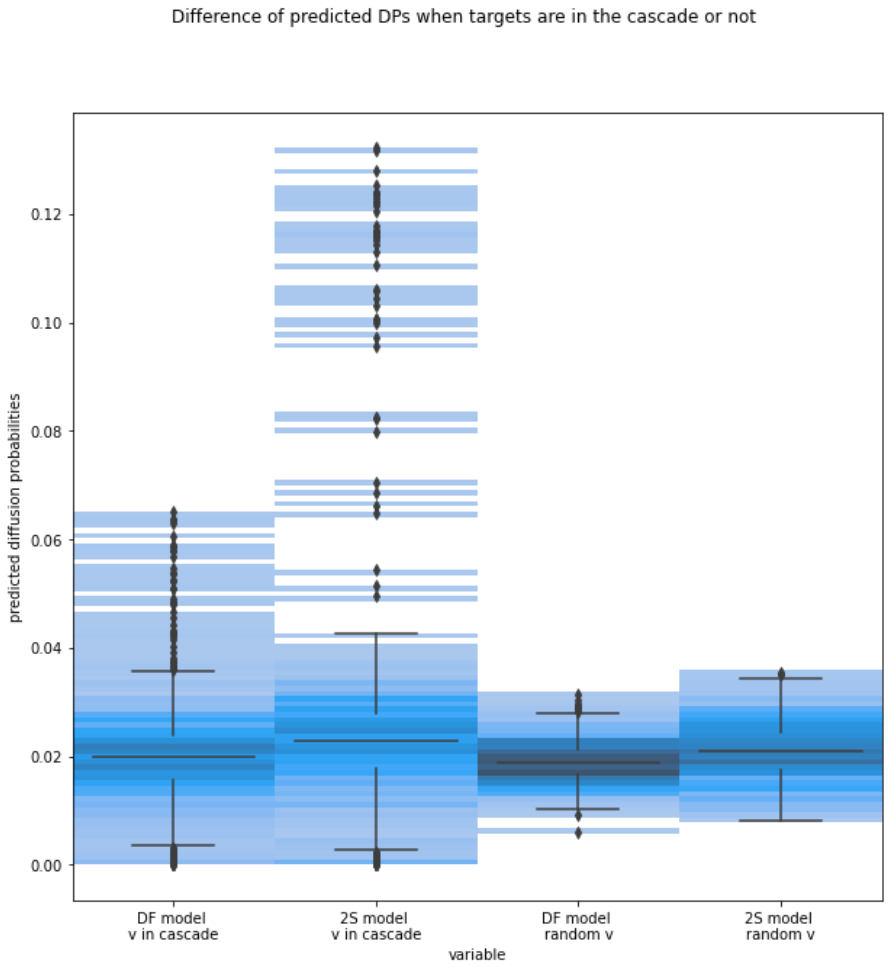


Results

Question : Can the models predict if a node v appears in a cascade of u ?
i.e. is the DP between u and v higher when v appears in a cascade of u ?

2-stage model and df model seem to predict high DP only when v appears in a cascade of u

However, the average predicted probabilities are not significantly higher



Results

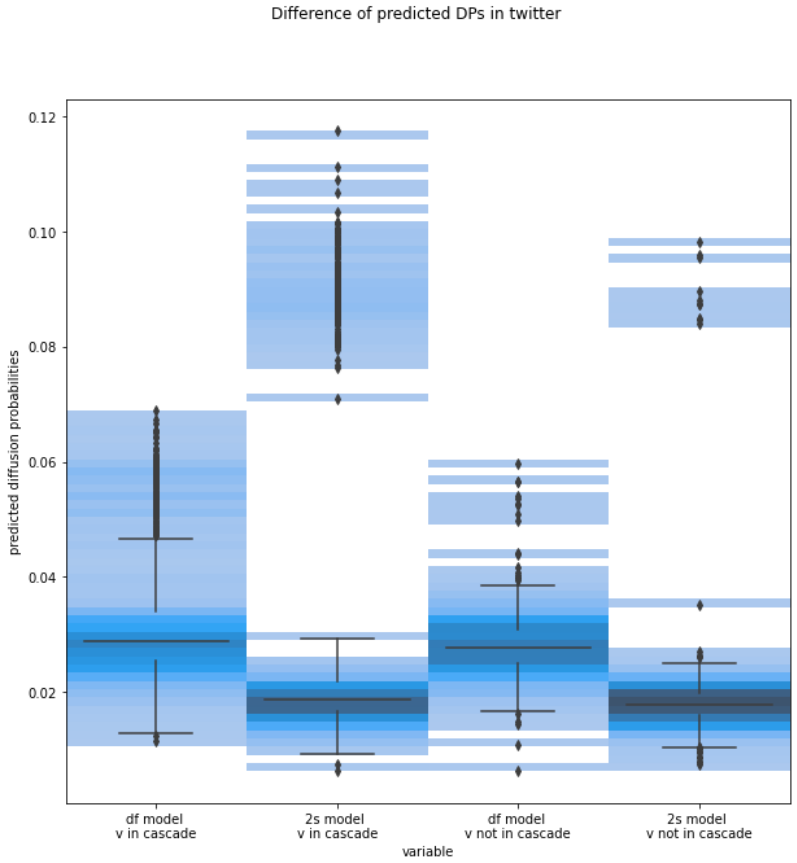
Question : Can the model be used on an other dataset ?

Test on the twitter dataset.
Only few features 6/22 :

- ingoing degree of i
- pagerank of i
- # of 2-reachable nodes from u
- outgoing degree of v
- pagerank of v
- presence of link between (u,v)

Others are set to 0

We only consider the influencers having more than 100 posts and targets reacting to more than 100 posts



Results : DNI/Expected spread on training data on Model 1

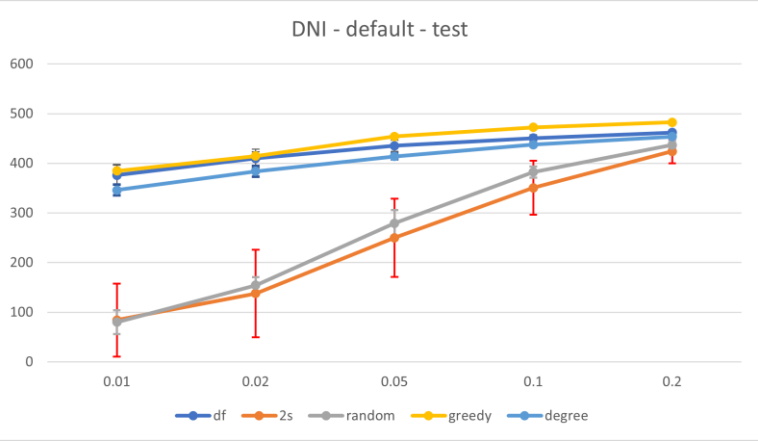
On dense dataset
tested on 20 instances of size 500x500

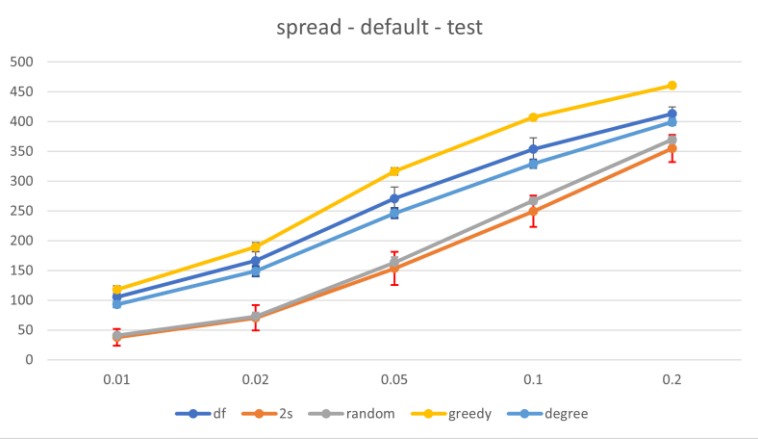
Model 1 :

- Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better than other methods for small k





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**376.22 (19.92)**	**409.7 (18.51)**	**435.2 (11.42)**	**450.58 (7.29)**	**461.92 (5.23)**
2s	83.97 (73.31)	137.93 (88.21)	249.95 (78.89)	350.55 (54.59)	424.07 (24.07)
random	79.93 (24.0)	154.25 (16.12)	279.22 (26.43)	382.33 (11.42)	436.72 (6.48)
greedy	384.85 (12.85)	414.73 (8.49)	454.2 (3.39)	472.57 (3.03)	482.55 (2.13)
degree	346.15 (10.67)	383.93 (10.83)	413.87 (6.12)	437.5 (4.22)	453.87 (4.11)

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**105.53 (10.29)**	**166.42 (14.96)**	**270.79 (19.2)**	**353.69 (19.09)**	**413.22 (11.44)**
2s	37.84 (13.91)	70.56 (21.35)	153.57 (27.82)	249.19 (26.2)	354.98 (22.77)
random	40.96 (3.18)	72.79 (6.5)	163.58 (8.91)	267.05 (4.73)	369.52 (5.51)
greedy	118.09 (6.55)	189.5 (7.45)	316.46 (6.14)	407.24 (4.06)	460.72 (2.72)
degree	92.97 (4.02)	148.75 (8.08)	246.21 (7.93)	329.03 (6.51)	399.04 (4.85)

Results : DNI/Expected spread on training data on Model 2 : INFECTOR

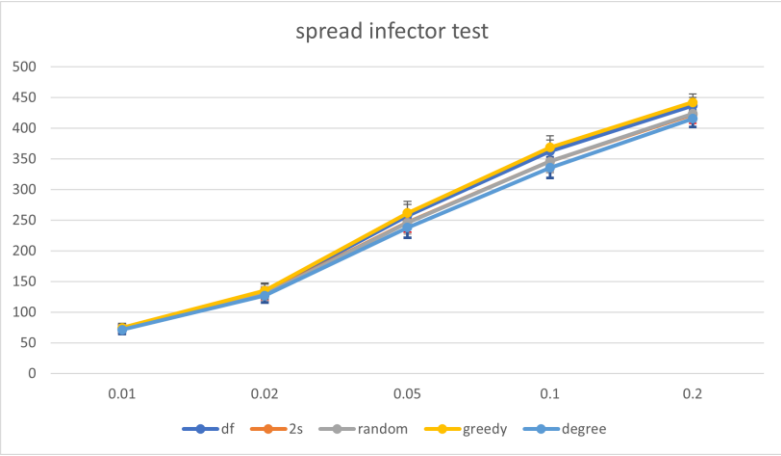
On dense dataset
tested on 20 instances of size 500x500

Model 2 :

- Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- INFECTOR labels

Observations :

- everything is as good as random



	Spread				
K/N	0.01	0.02	0.05	0.1	0.2
df	**75.06 (1.88)**	**136.27 (3.21)**	**262.3 (5.32)**	**367.47 (6.09)**	**440.77 (4.93)**
2s	74.04 (2.47)	133.04 (5.02)	250.92 (9.92)	350.83 (11.03)	426.58 (7.71)
random	73.71 (1.79)	132.31 (2.98)	249.48 (4.34)	350.41 (4.49)	427.5 (3.41)
greedy	76.17 (1.86)	138.2 (3.1)	266.68 (4.74)	374.02 (4.78)	446.19 (3.4)
degree	72.82 (1.75)	129.68 (2.82)	241.95 (4.1)	339.86 (4.26)	419.19 (3.35)

Results : DNI/Expected spread on training data on Model 3 : no mapping

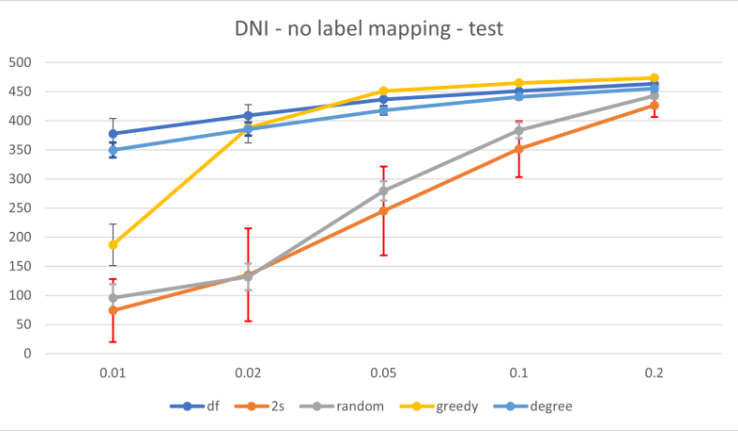
On dense dataset
tested on 20 instances of size 500x500

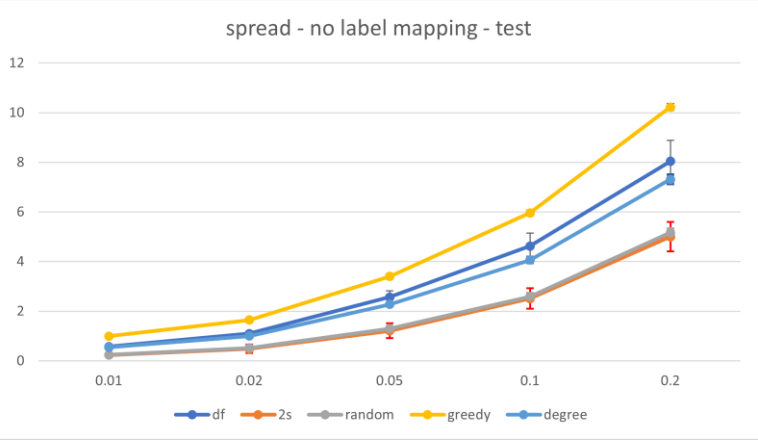
Model 3 :

- Aggregated cascade features
- Regularization to avoid local minima
- NO Mapping of label on higher values
- data-based labels

Observations :

- greedy gives very bad results for low K
- DF performs better than other methods
- 2s has very large std





DNI					
K/N	0.01	0.02	0.05	0.1	0.2
df	**382.78 (17.11)**	**411.31 (17.9)**	**437.51 (12.92)**	**451.18 (8.98)**	**463.55 (5.05)**
2s	77.0 (55.59)	139.53 (70.48)	248.55 (73.58)	347.95 (48.94)	424.83 (22.02)
Random	82.11 (14.53)	143.33 (12.95)	285.27 (9.19)	378.63 (7.85)	439.25 (2.53)
greedy	169.93 (8.91)	371.22 (6.38)	452.88 (0.64)	464.9 (0.59)	473.88 (0.52)
degree	354.83 (3.21)	386.83 (2.78)	415.97 (1.85)	440.28 (1.02)	455.02 (0.95)

Spread					
K/N	0.01	0.02	0.05	0.1	0.2
Df	**0.58 (0.03)**	**1.11 (0.1)**	**2.54 (0.26)**	**4.64 (0.48)**	**7.98 (0.74)**
2s	0.24 (0.1)	0.48 (0.15)	1.22 (0.27)	2.46 (0.41)	4.95 (0.58)
random	0.25 (0.02)	0.5 (0.02)	1.28 (0.04)	2.55 (0.06)	5.12 (0.06)
greedy	1.02 (0.02)	1.68 (0.02)	3.42 (0.03)	5.95 (0.03)	10.18 (0.04)
degree	0.53 (0.01)	0.99 (0.01)	2.25 (0.02)	4.02 (0.03)	7.23 (0.05)

Results : DNI/Expected spread on training data on Model 4 : No regularization

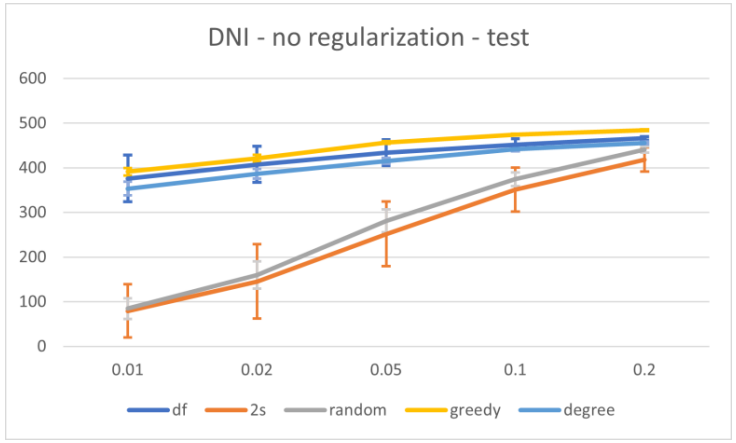
On dense dataset
tested on 20 instances of size 500x500

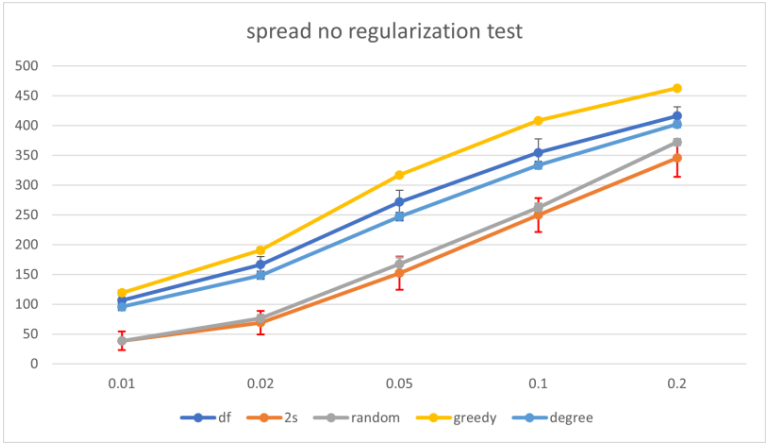
Model 4 :

- Aggregated cascade features
- NO Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is still better in average but with higher std





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**376.0 (52.19)**	**407.45 (40.29)**	**433.65 (29.19)**	**451.97 (13.08)**	**466.02 (4.19)**
2s	79.45 (59.42)	145.47 (83.28)	251.97 (72.27)	350.67 (49.34)	418.37 (27.19)
random	84.65 (23.04)	160.05 (30.34)	281.22 (25.88)	374.43 (15.19)	440.65 (7.07)
greedy	391.08 (8.52)	420.75 (7.88)	456.27 (2.9)	474.25 (2.29)	484.28 (1.9)
degree	353.5 (14.92)	386.25 (10.76)	415.53 (6.28)	441.35 (4.23)	455.88 (3.02)

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**105.53 (10.29)**	**166.42 (14.96)**	**270.79 (19.2)**	**353.69 (19.09)**	**413.22 (11.44)**
2s	37.84 (13.91)	70.56 (21.35)	153.57 (27.82)	249.19 (26.2)	354.98 (22.77)
random	40.96 (3.18)	72.79 (6.5)	163.58 (8.91)	267.05 (4.73)	369.52 (5.51)
greedy	118.09 (6.55)	189.5 (7.45)	316.46 (6.14)	407.24 (4.06)	460.72 (2.72)
degree	92.97 (4.02)	148.75 (8.08)	246.21 (7.93)	329.03 (6.51)	399.04 (4.85)

Results : DNI/Expected spread on training data on Model 5 : Without cascade features

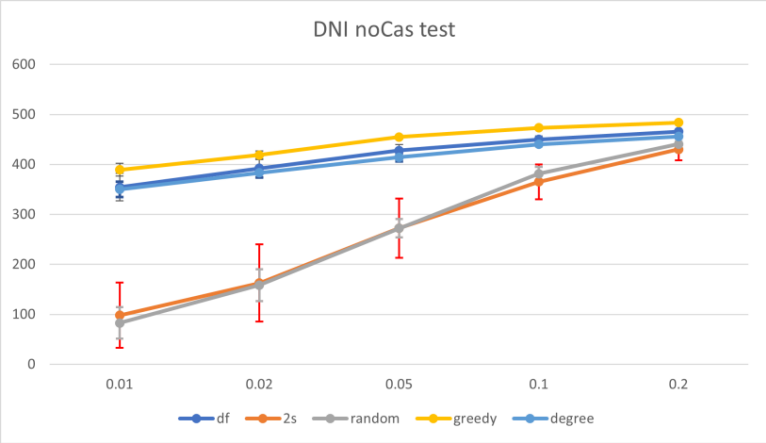
On dense dataset
tested on 20 instances of size 500x500

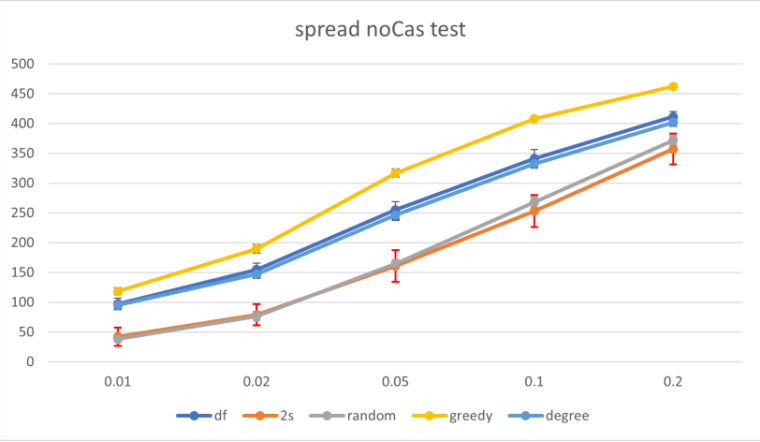
Model 5 :

- NO Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF performs as good as degree heuristic





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**354.37 (27.5)**	**392.22 (17.75)**	**427.83 (11.79)**	**449.92 (6.87)**	**465.6 (4.91)**
2s	98.02 (65.12)	162.55 (77.18)	272.3 (59.61)	365.15 (35.03)	430.07 (22.11)
random	82.62 (31.61)	158.27 (31.75)	272.17 (18.49)	381.25 (13.49)	440.43 (4.58)
greedy	389.18 (12.72)	418.88 (7.84)	454.87 (3.79)	473.25 (3.17)	483.53 (2.34)
degree	350.18 (15.53)	383.3 (9.65)	414.28 (8.08)	440.33 (4.01)	455.8 (4.18)

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**97.2 (9.78)**	**154.71 (10.95)**	**255.24 (13.53)**	**341.19 (15.03)**	**411.38 (8.62)**
2s	42.21 (15.13)	79.03 (17.81)	160.67 (26.59)	252.97 (26.91)	357.07 (26.0)
random	38.26 (4.03)	76.64 (7.28)	164.98 (6.05)	267.95 (8.61)	371.79 (7.87)
greedy	118.39 (5.95)	189.76 (7.05)	316.78 (6.92)	407.96 (4.97)	462.16 (3.21)
degree	95.0 (5.16)	147.15 (6.14)	246.54 (7.85)	332.53 (6.23)	401.93 (5.49)

Results : DNI/Expected spread on sparse instance on Model 1

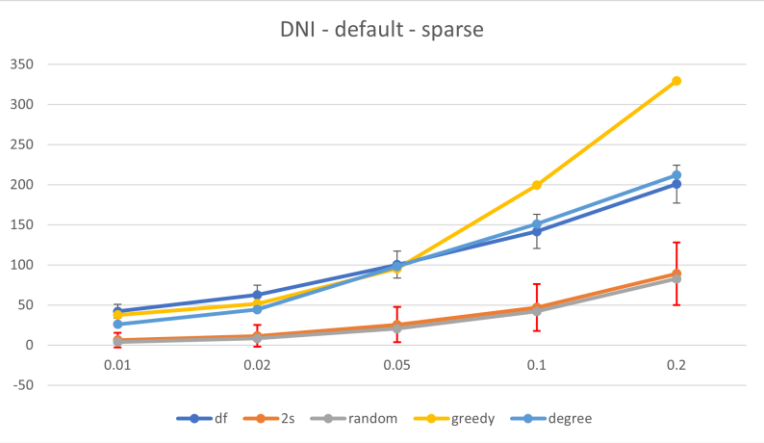
On sparse dataset
tested on 20 instances of size 1000x1000

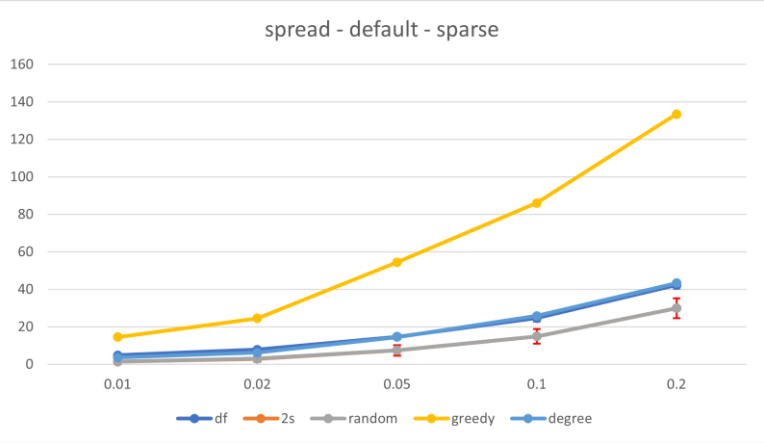
Model 1 :

- Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better for low K for DNI
- For spread, DF performs as good as degree





DNI					
K/N	0.01	0.02	0.05	0.1	0.2
df	**42.21 (8.7)**	**62.7 (11.81)**	**100.3 (16.91)**	141.83 (21.45)	200.78 (23.59)
2s	6.3 (8.98)	11.54 (13.64)	25.55 (21.89)	47.13 (29.22)	89.08 (39.13)
random	4.03 (0.55)	8.51 (0.97)	20.74 (1.94)	42.03 (1.98)	82.82 (3.35)
greedy	37.65 (0.0)	51.75 (0.0)	95.45 (0.0)	199.4 (0.0)	329.45 (0.0)
degree	26.1 (0.0)	44.35 (0.0)	98.15 (0.0)	**151.15 (0.0)**	**212.0 (0.0)**

Spread					
K/N	0.01	0.02	0.05	0.1	0.2
df	**4.8 (0.73)**	**7.81 (1.03)**	**14.71 (1.43)**	24.64 (2.04)	**4.8 (0.73)**
2s	1.45 (0.94)	2.89 (1.56)	7.39 (2.86)	14.91 (3.87)	1.45 (0.94)
random	1.55 (0.17)	3.01 (0.23)	7.54 (0.41)	14.9 (0.51)	1.55 (0.17)
greedy	14.49 (0.0)	24.49 (0.0)	54.41 (0.0)	86.08 (0.0)	14.49 (0.0)
degree	3.61 (0.0)	6.18 (0.0)	14.58 (0.0)	**25.79 (0.0)**	3.61 (0.0)

Results : DNI/Expected spread on sparse data on Model 3 : no mapping

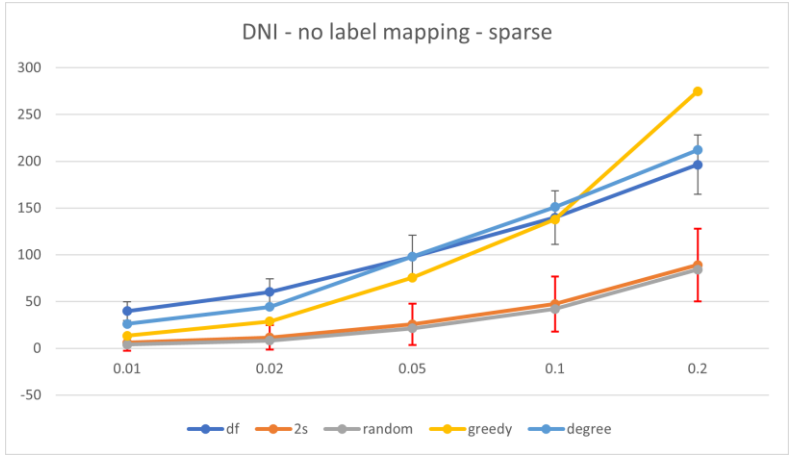
On sparse dataset
tested on 20 instances of size 1000x1000

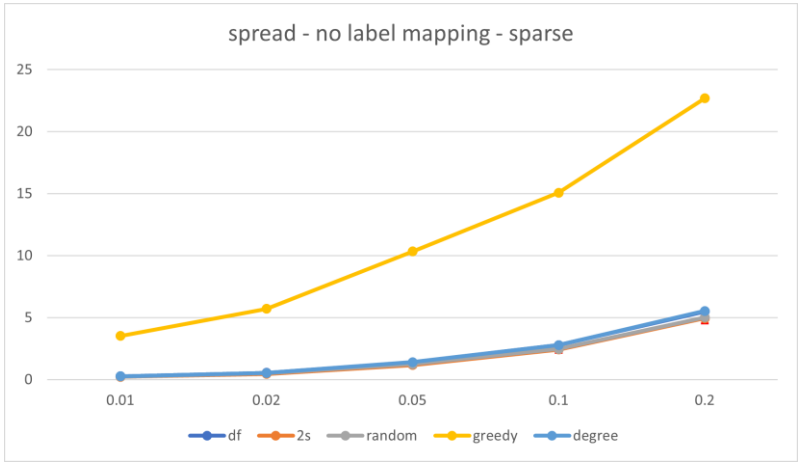
Model 3 :

- Aggregated cascade features
- Regularization to avoid local minima
- NO Mapping of label on higher values
- data-based labels

Observations :

- Spreads are close to random
- df and degree are better than greedy for low k





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**39.72 (9.9)**	**60.19 (13.92)**	97.99 (22.75)	139.8 (28.74)	196.35 (31.84)
2s	6.01 (8.58)	11.65 (13.2)	25.61 (21.92)	47.38 (29.47)	89.15 (38.88)
random	4.07 (0.65)	8.23 (0.85)	21.39 (1.41)	42.16 (2.35)	84.17 (1.94)
greedy	13.5 (0.0)	28.75 (0.0)	75.65 (0.0)	137.7 (0.0)	274.95 (0.0)
degree	26.1 (0.0)	44.35 (0.0)	**98.15 (0.0)**	**151.15 (0.0)**	**212.0 (0.0)**

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**0.27 (0.02)**	**0.55 (0.01)**	1.38 (0.05)	2.78 (0.1)	5.5 (0.17)
2s	0.22 (0.05)	0.45 (0.09)	1.18 (0.18)	2.42 (0.27)	4.95 (0.42)
random	0.25 (0.04)	0.52 (0.04)	1.26 (0.09)	2.47 (0.15)	5.02 (0.13)
greedy	3.52 (0.0)	5.7 (0.0)	10.34 (0.0)	15.08 (0.0)	22.69 (0.0)
degree	**0.27 (0.0)**	0.53 (0.0)	**1.41 (0.0)**	**2.79 (0.0)**	**5.54 (0.0)**

Results : DNI/Expected spread on sparse data on Model 4 : No regularization

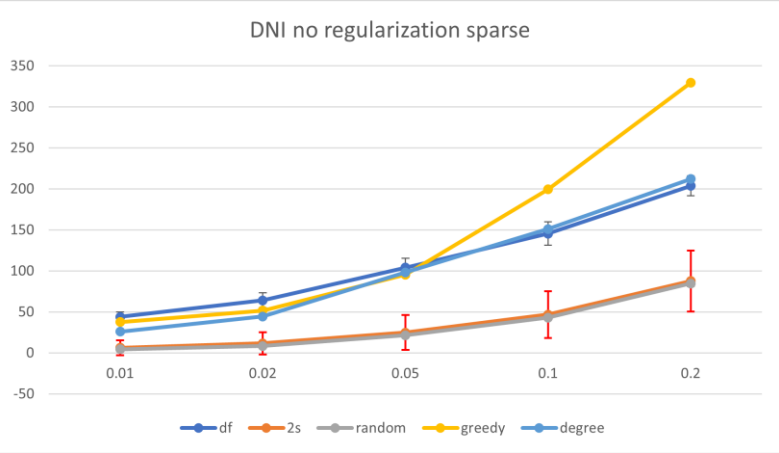
On sparse dataset
tested on 20 instances of size 1000x1000

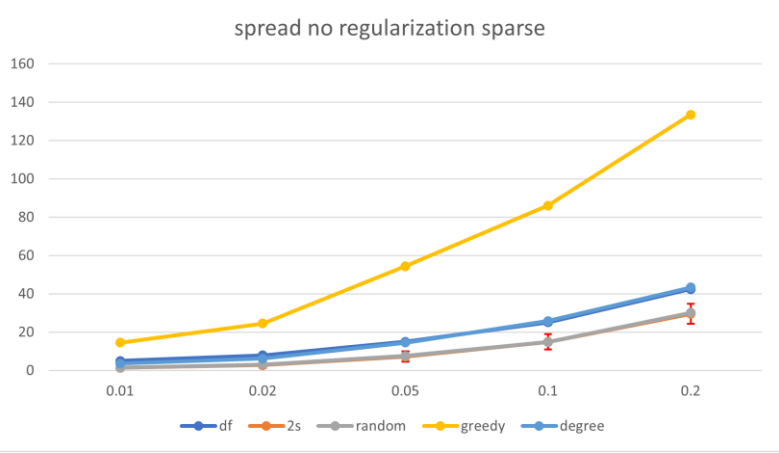
Model 4 :

- Aggregated cascade features
- NO Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better for lower K





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**43.9 (6.26)**	**64.15 (9.0)**	**104.13 (11.51)**	145.54 (14.24)	203.68 (12.25)
2s	6.3 (9.26)	11.66 (13.44)	24.9 (21.32)	46.88 (28.5)	87.67 (37.13)
random	4.45 (0.62)	8.37 (0.96)	21.59 (1.7)	42.98 (2.05)	84.71 (3.32)
greedy	37.65 (0.0)	51.75 (0.0)	95.45 (0.0)	199.4 (0.0)	329.45 (0.0)
degree	26.1 (0.0)	44.35 (0.0)	98.15 (0.0)	**151.15 (0.0)**	**212.0 (0.0)**

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**5.02 (0.52)**	**7.92 (0.68)**	**15.03 (1.09)**	25.1 (1.48)	42.49 (1.09)
2s	1.47 (0.97)	2.87 (1.53)	7.22 (2.68)	14.97 (3.97)	29.58 (5.13)
random	1.44 (0.16)	3.11 (0.34)	7.6 (0.41)	14.79 (0.59)	30.13 (0.98)
greedy	14.49 (0.0)	24.49 (0.0)	54.41 (0.0)	86.08 (0.0)	133.45 (0.0)
degree	3.61 (0.0)	6.18 (0.0)	14.58 (0.0)	**25.79 (0.0)**	**43.31 (0.0)**

Results : DNI/Expected spread on sparse data on Model 5 : Without cascade features

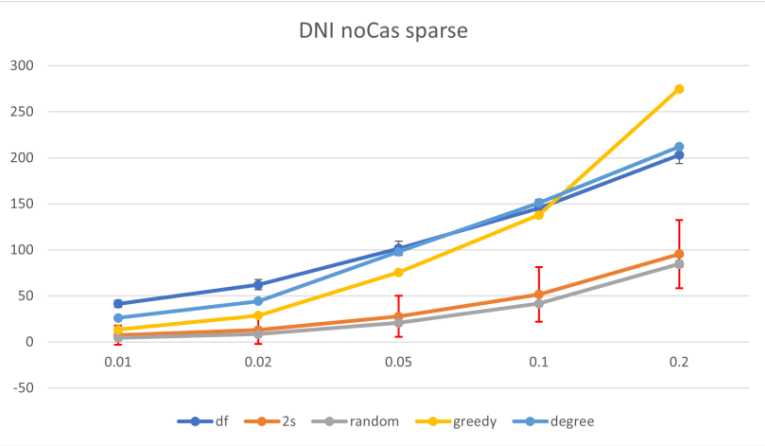
On sparse dataset
tested on 20 instances of size 1000x1000

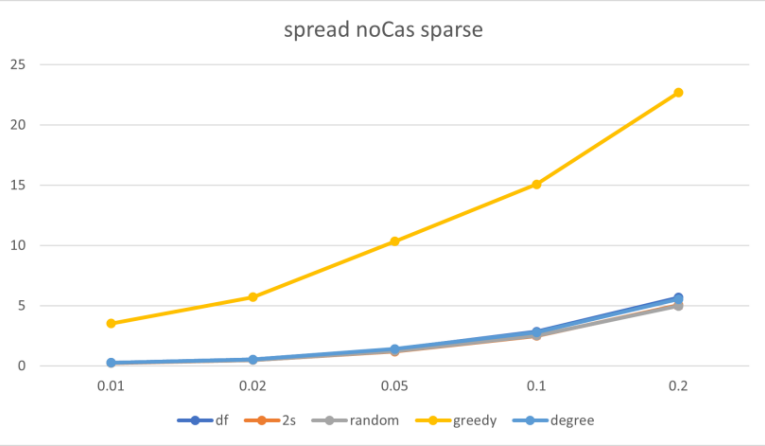
Model 5 :

- NO Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better for lower K
- Spread are close to random





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**41.51 (3.84)**	**62.12 (5.46)**	**101.57 (7.84)**	145.3 (9.41)	203.03 (9.28)
2s	7.36 (10.3)	13.1 (15.13)	27.81 (22.37)	51.63 (29.77)	95.42 (37.02)
random	4.35 (0.5)	8.5 (0.82)	20.83 (1.83)	41.65 (2.02)	84.7 (3.39)
greedy	13.5 (0.0)	28.75 (0.0)	75.65 (0.0)	137.7 (0.0)	274.95 (0.0)
degree	26.1 (0.0)	44.35 (0.0)	98.15 (0.0)	**151.15 (0.0)**	**212.0 (0.0)**

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**0.27 (0.01)**	**0.53 (0.02)**	1.39 (0.03)	**2.86 (0.08)**	**5.68 (0.12)**
2s	0.24 (0.06)	0.5 (0.08)	1.2 (0.18)	2.48 (0.2)	5.06 (0.31)
random	0.26 (0.04)	0.48 (0.06)	1.23 (0.08)	2.52 (0.11)	4.98 (0.19)
greedy	3.52 (0.0)	5.7 (0.0)	10.34 (0.0)	15.08 (0.0)	22.69 (0.0)
degree	**0.27 (0.0)**	**0.53 (0.0)**	**1.41 (0.0)**	2.79 (0.0)	5.54 (0.0)

Results : DNI/Expected spread on twitter instance on Model 1

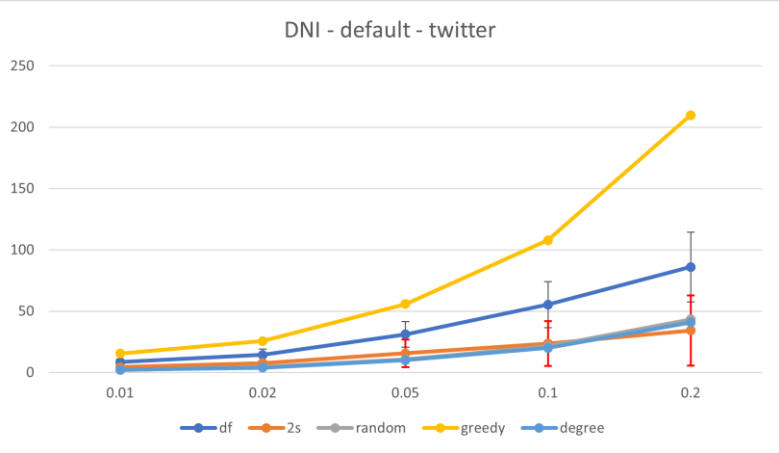
On twitter dataset
tested on 20 instances of size 1000x1000

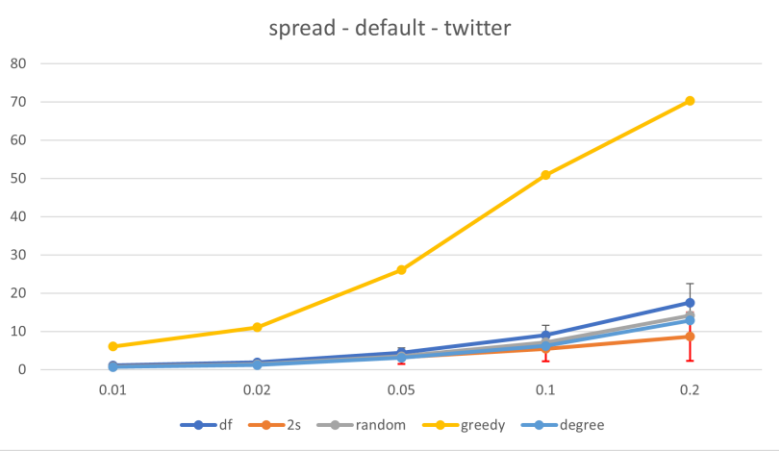
Model 1 :

- Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better than others





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**8.46 (2.36)**	**14.47 (4.6)**	**31.08 (10.36)**	**55.34 (18.87)**	**85.95 (28.61)**
2s	4.42 (2.66)	7.75 (5.03)	15.65 (11.25)	23.69 (18.27)	34.25 (28.69)
random	2.06 (0.36)	4.36 (0.42)	10.69 (0.76)	21.42 (0.99)	43.28 (1.11)
greedy	15.5 (0.0)	25.75 (0.0)	55.85 (0.0)	107.9 (0.0)	209.8 (0.0)
degree	2.3 (0.0)	4.0 (0.0)	10.05 (0.0)	20.05 (0.0)	40.8 (0.0)

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**1.09 (0.23)**	**1.9 (0.47)**	**4.47 (1.2)**	**9.04 (2.55)**	**17.54 (4.96)**
2s	0.8 (0.27)	1.42 (0.64)	3.23 (1.75)	5.47 (3.33)	8.66 (6.32)
random	0.69 (0.09)	1.41 (0.22)	3.53 (0.15)	7.17 (0.52)	14.18 (0.5)
greedy	6.06 (0.0)	11.07 (0.0)	26.07 (0.0)	50.87 (0.0)	70.29 (0.0)
degree	0.71 (0.0)	1.24 (0.0)	3.1 (0.0)	6.21 (0.0)	12.88 (0.0)

Results : DNI/Expected spread on twitter data on Model 3 : no mapping

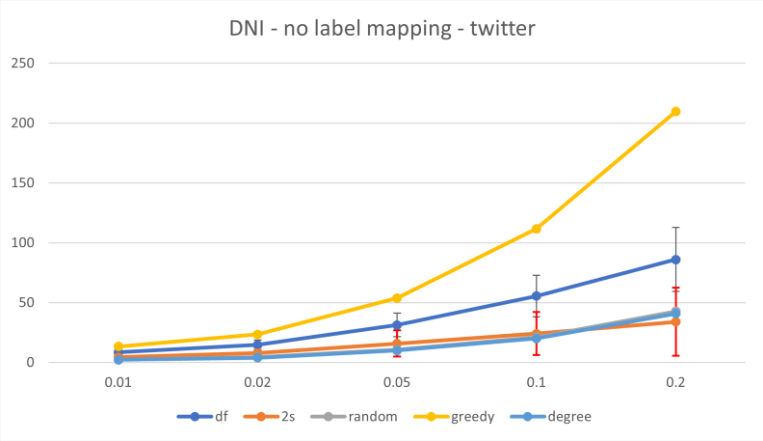
On twitter dataset
tested on 20 instances of size 1000x1000

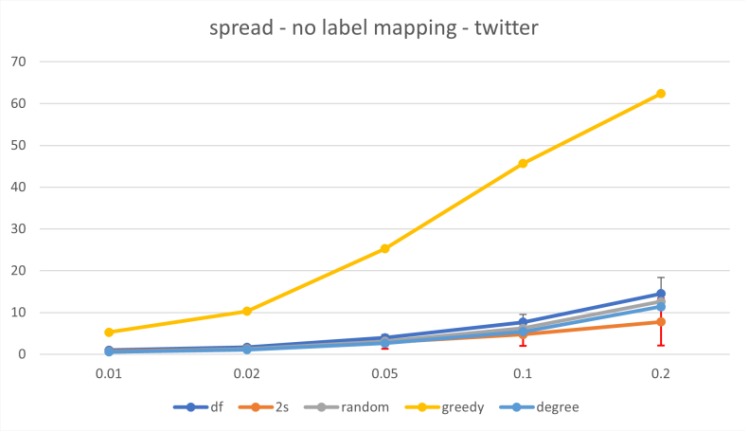
Model 3 :

- Aggregated cascade features
- Regularization to avoid local minima
- NO Mapping of label on higher values
- data-based labels

Observations :

- DF is better than others





DNI					
K/N	0.01	0.02	0.05	0.1	0.2
df	**8.51 (2.35)**	**14.74 (4.11)**	**31.28 (9.89)**	**55.52 (17.28)**	**86.1 (26.6)**
2s	4.41 (2.62)	7.85 (5.17)	15.86 (11.03)	24.15 (18.06)	34.01 (28.43)
random	2.3 (0.21)	4.39 (0.5)	10.7 (0.8)	21.12 (0.9)	42.73 (1.71)
greedy	13.35 (0.0)	23.5 (0.0)	53.75 (0.0)	111.7 (0.0)	209.75 (0.0)
degree	2.3 (0.0)	4.0 (0.0)	10.05 (0.0)	20.05 (0.0)	40.8 (0.0)

Spread					
K/N	0.01	0.02	0.05	0.1	0.2
df	**0.96 (0.19)**	**1.69 (0.32)**	**3.93 (0.83)**	**7.66 (1.9)**	**14.5 (3.84)**
2s	0.69 (0.21)	1.28 (0.52)	2.79 (1.47)	4.74 (2.76)	7.75 (5.64)
random	0.6 (0.11)	1.22 (0.17)	3.1 (0.18)	6.21 (0.3)	12.63 (0.27)
greedy	5.28 (0.0)	10.28 (0.0)	25.28 (0.0)	45.68 (0.0)	62.4 (0.0)
degree	0.59 (0.0)	1.1 (0.0)	2.64 (0.0)	5.4 (0.0)	11.4 (0.0)

Results : DNI/Expected spread on twitter data on Model 4 : No regularization

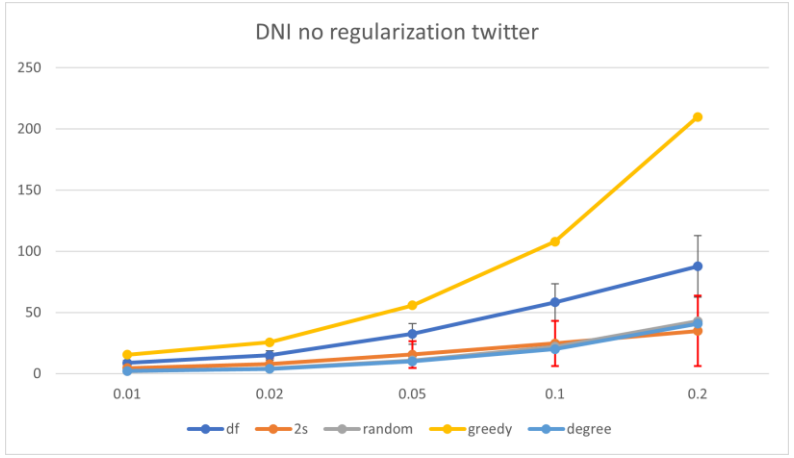
On twitter dataset
tested on 20 instances of size 1000x1000

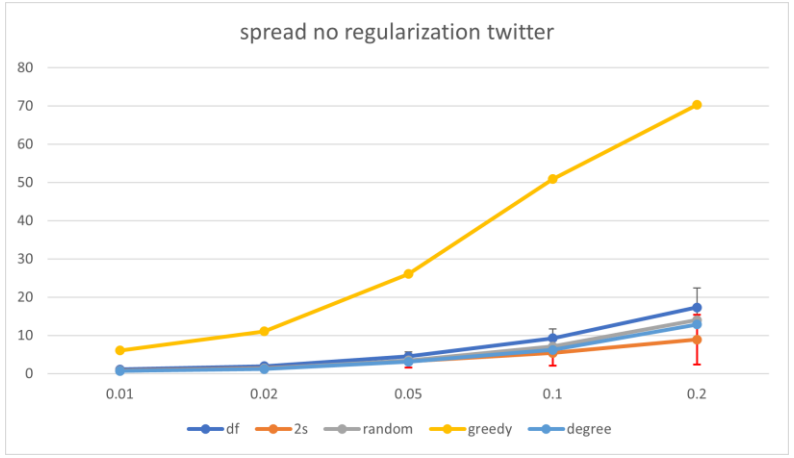
Model 4 :

- Aggregated cascade features
- NO Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- DF is better for lower K
- DF has higher std





DNI

K/N	0.01	0.02	0.05	0.1	0.2
df	**8.74 (2.07)**	**15.12 (3.75)**	**32.61 (8.44)**	**58.28 (15.3)**	**87.71 (25.07)**
2s	4.56 (2.66)	7.84 (4.94)	15.67 (10.92)	24.72 (18.51)	34.95 (28.82)
random	2.08 (0.31)	4.17 (0.5)	10.76 (0.89)	22.05 (0.95)	42.84 (1.26)
greedy	15.5 (0.0)	25.75 (0.0)	55.85 (0.0)	107.9 (0.0)	209.8 (0.0)
degree	2.3 (0.0)	4.0 (0.0)	10.05 (0.0)	20.05 (0.0)	40.8 (0.0)

Spread

K/N	0.01	0.02	0.05	0.1	0.2
df	**1.1 (0.24)**	**1.94 (0.47)**	**4.52 (1.2)**	**9.25 (2.44)**	**17.34 (5.04)**
2s	0.79 (0.27)	1.43 (0.6)	3.22 (1.67)	5.41 (3.29)	8.91 (6.47)
random	0.77 (0.09)	1.36 (0.13)	3.45 (0.26)	7.1 (0.28)	14.0 (0.43)
greedy	6.06 (0.0)	11.07 (0.0)	26.07 (0.0)	50.87 (0.0)	70.29 (0.0)
degree	0.71 (0.0)	1.24 (0.0)	3.1 (0.0)	6.21 (0.0)	12.88 (0.0)

Results : DNI/Expected spread on twitter data on Model 5 : Without cascade features

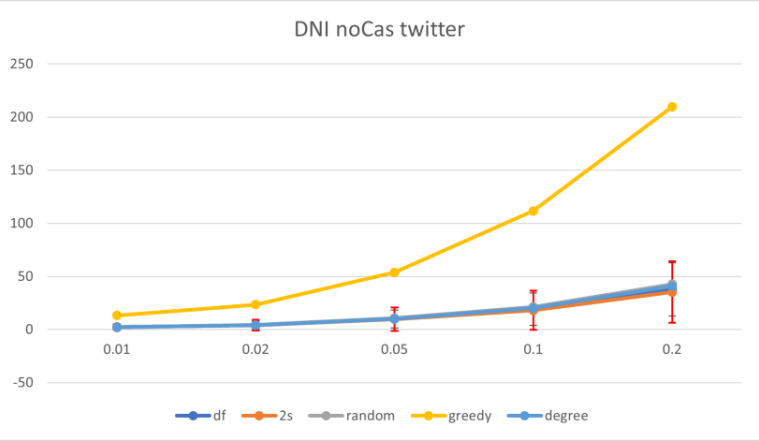
On twitter dataset
tested on 20 instances of size 1000x1000

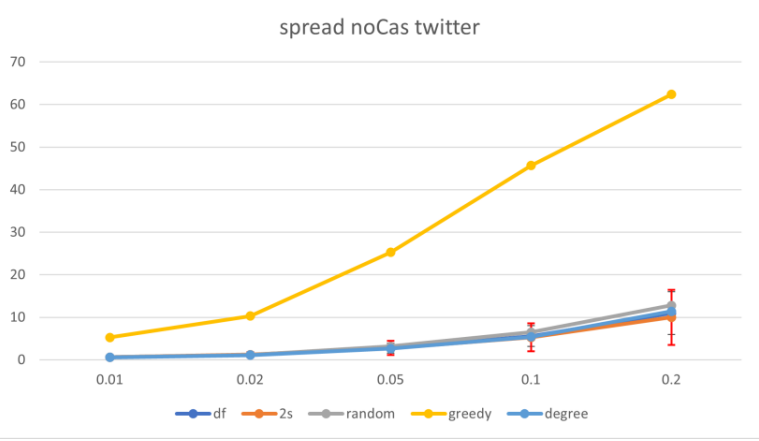
Model 5 :

- NO Aggregated cascade features
- Regularization to avoid local minima
- Mapping of label on higher values
- data-based labels

Observations :

- Spread and DNI are close to random





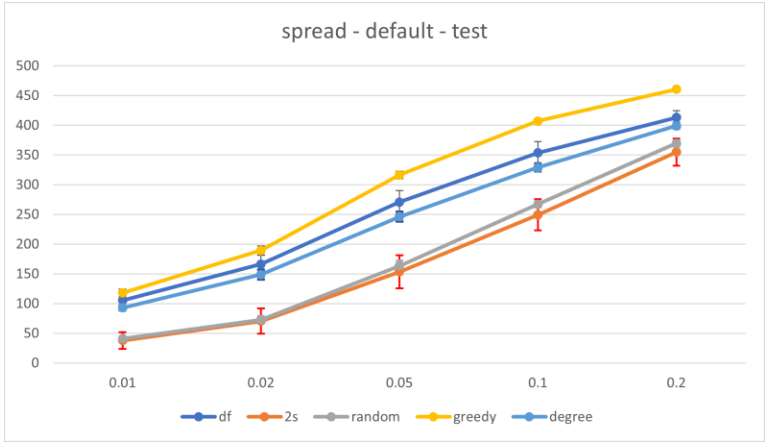
DNI

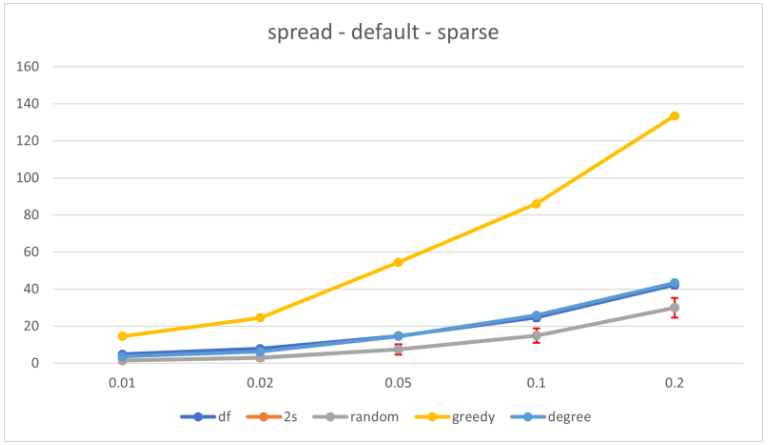
K/N	0.01	0.02	0.05	0.1	0.2
df	**2.49 (0.86)**	**4.16 (1.52)**	9.94 (3.8)	19.17 (7.41)	37.61 (14.65)
2s	2.36 (0.61)	4.1 (1.15)	9.73 (2.97)	18.11 (6.95)	35.28 (15.73)
random	1.99 (0.32)	4.33 (0.38)	10.9 (0.67)	21.33 (0.96)	42.8 (1.3)
greedy	13.35 (0.0)	23.5 (0.0)	53.75 (0.0)	111.7 (0.0)	209.75 (0.0)
degree	2.3 (0.0)	4.0 (0.0)	**10.05 (0.0)**	**20.05 (0.0)**	**40.8 (0.0)**

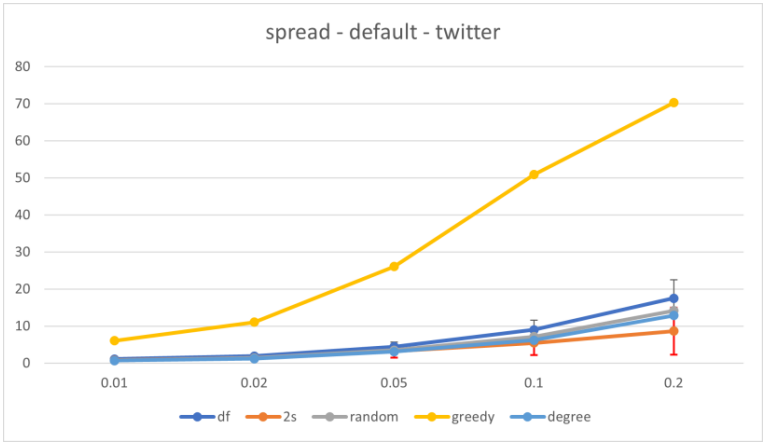
Spread

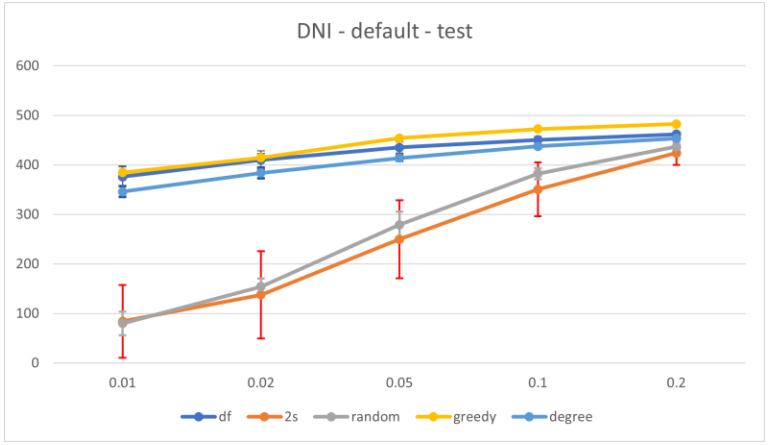
K/N	0.01	0.02	0.05	0.1	0.2
df	**0.63 (0.24)**	**1.14 (0.43)**	**2.75 (1.06)**	**5.58 (2.19)**	10.96 (4.28)
2s	0.62 (0.17)	1.22 (0.33)	2.77 (0.9)	5.26 (2.01)	10.0 (4.49)
random	0.62 (0.1)	1.21 (0.12)	3.15 (0.24)	6.47 (0.28)	12.78 (0.46)
greedy	5.28 (0.0)	10.28 (0.0)	25.28 (0.0)	45.68 (0.0)	62.4 (0.0)
degree	0.59 (0.0)	1.1 (0.0)	2.64 (0.0)	5.4 (0.0)	**11.4 (0.0)**

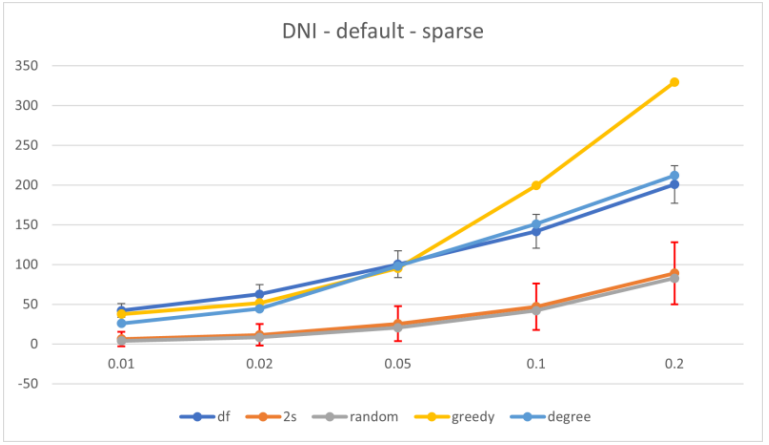
Results : Model 1

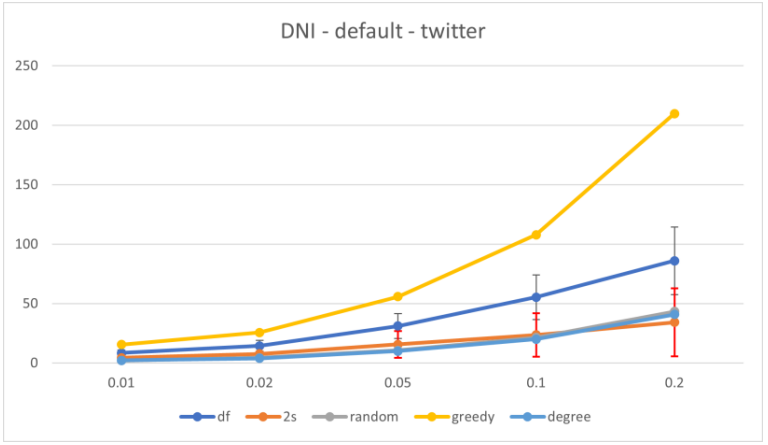




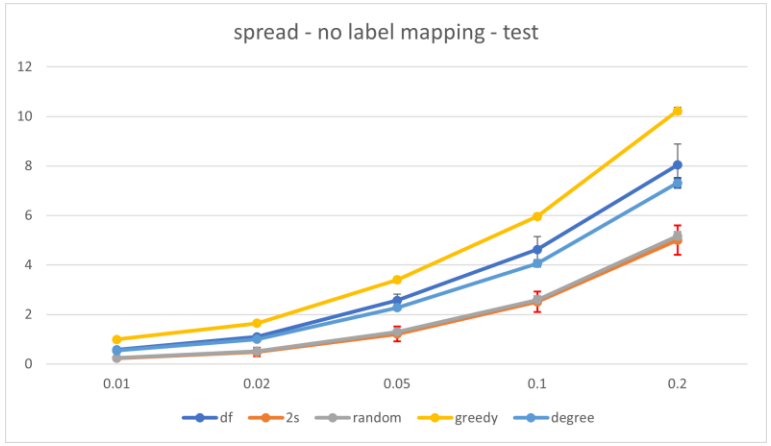


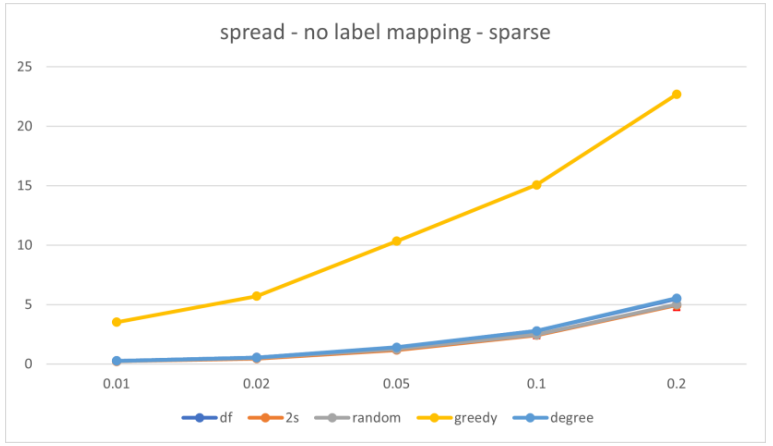


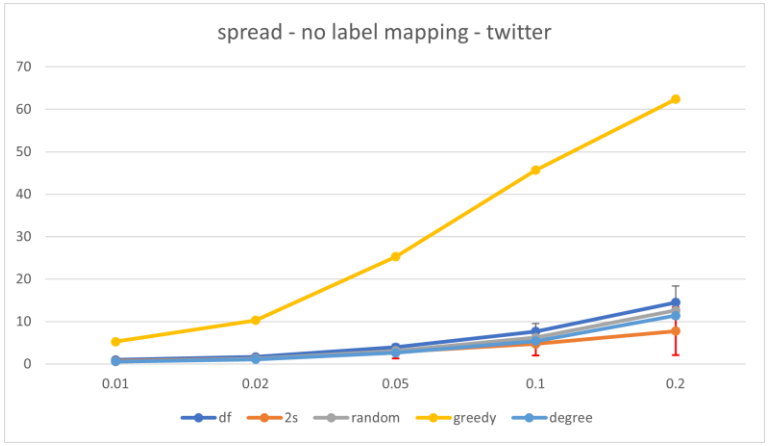


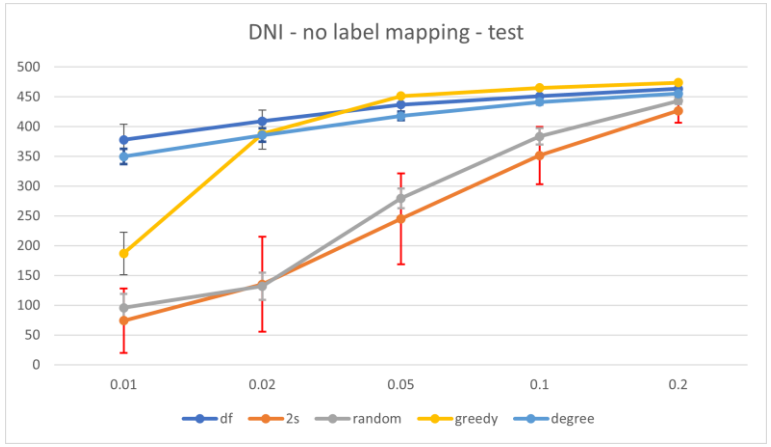


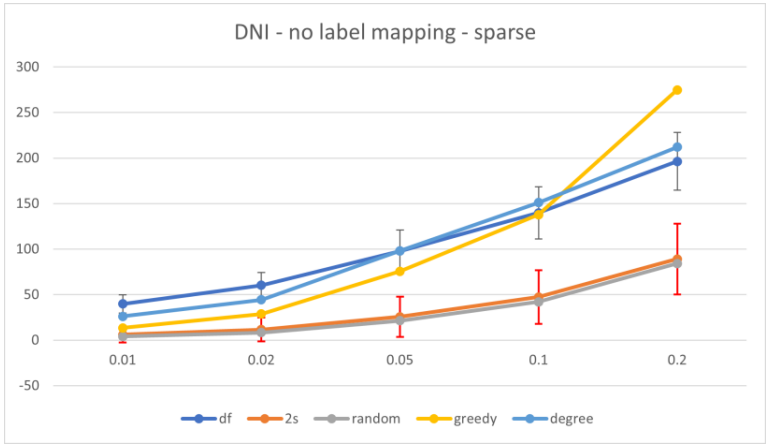
Results : Model 3 : no mapping

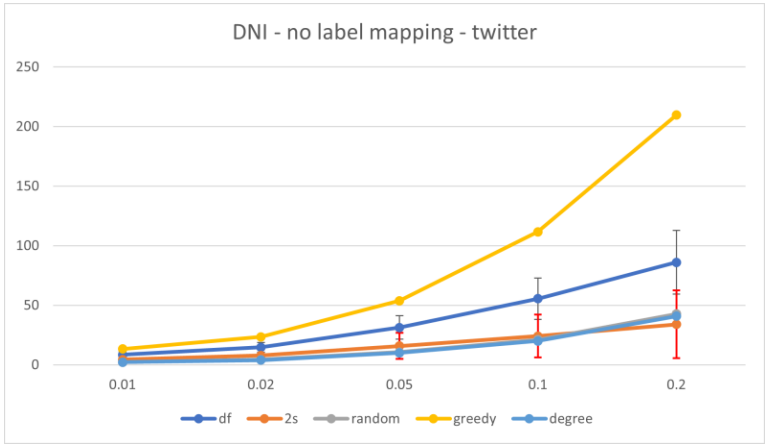




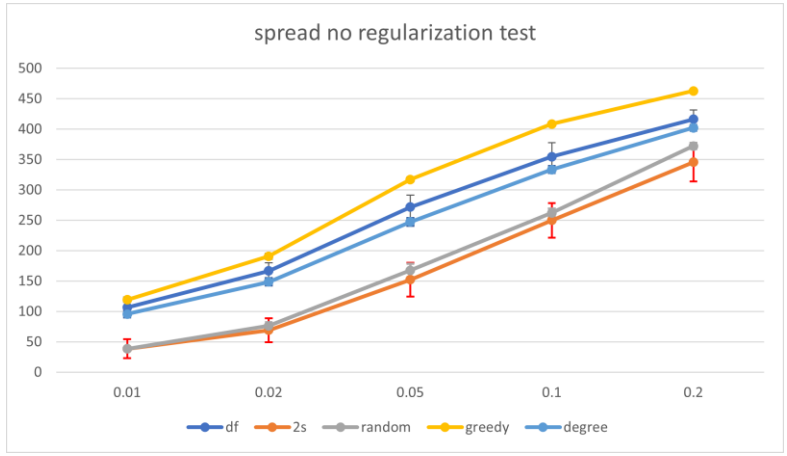


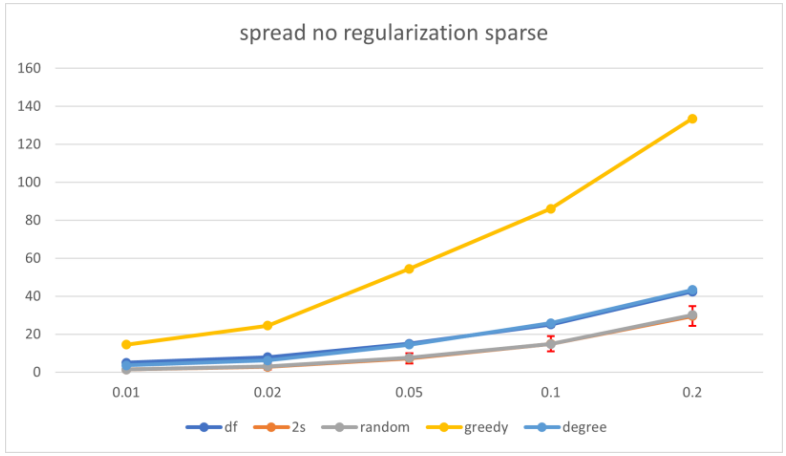


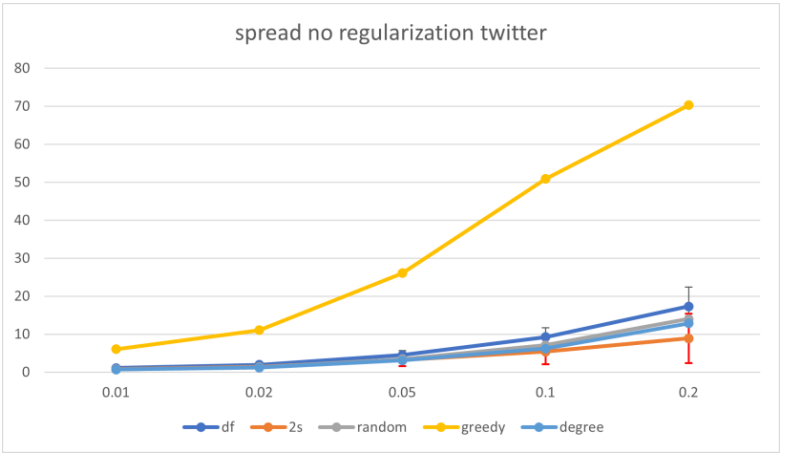


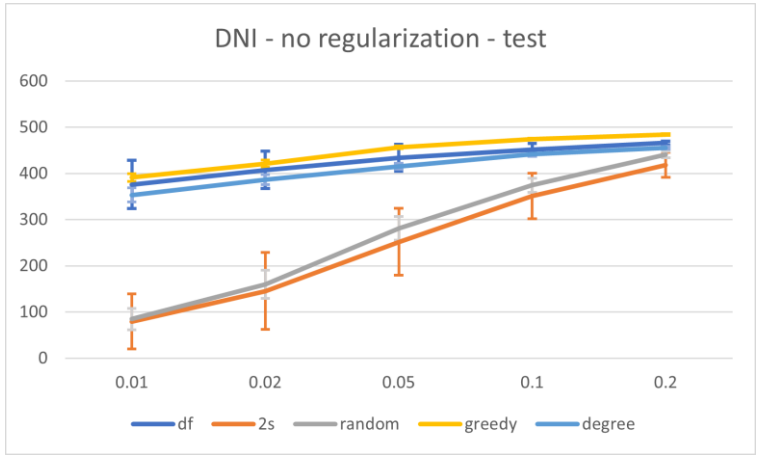


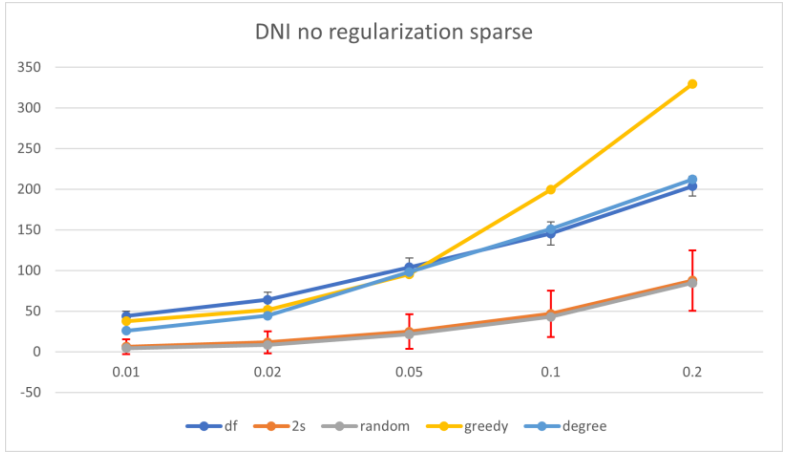
Results : Model 4 : no regularization

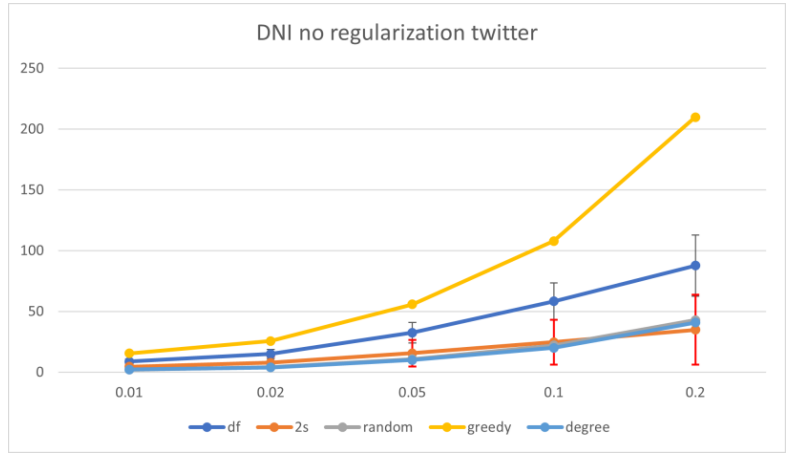




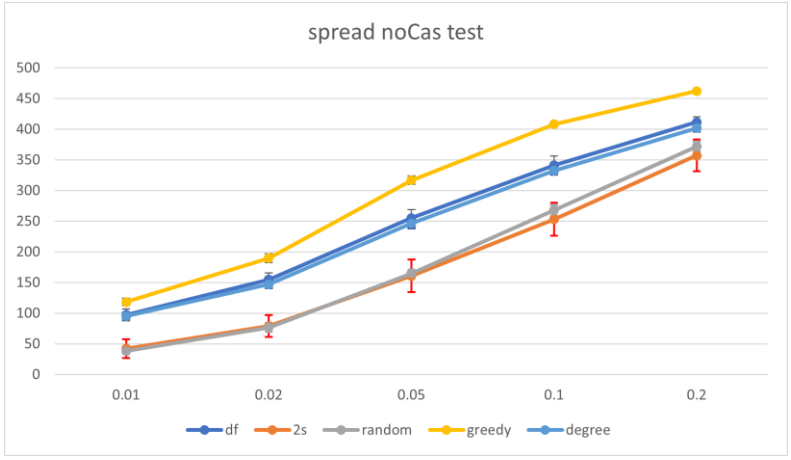


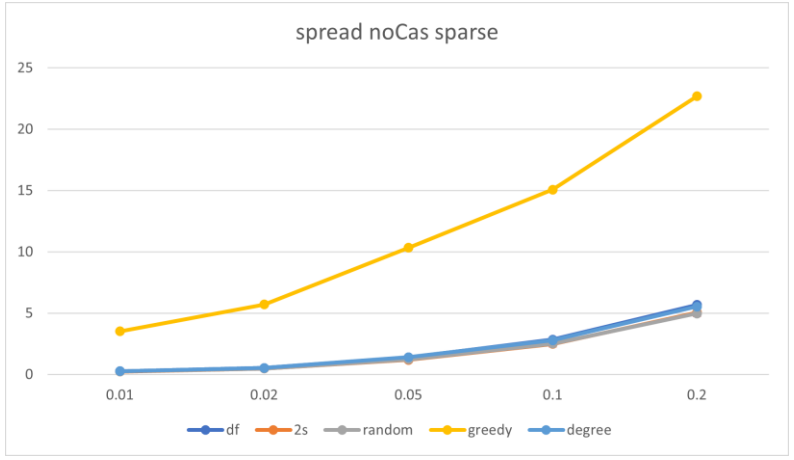


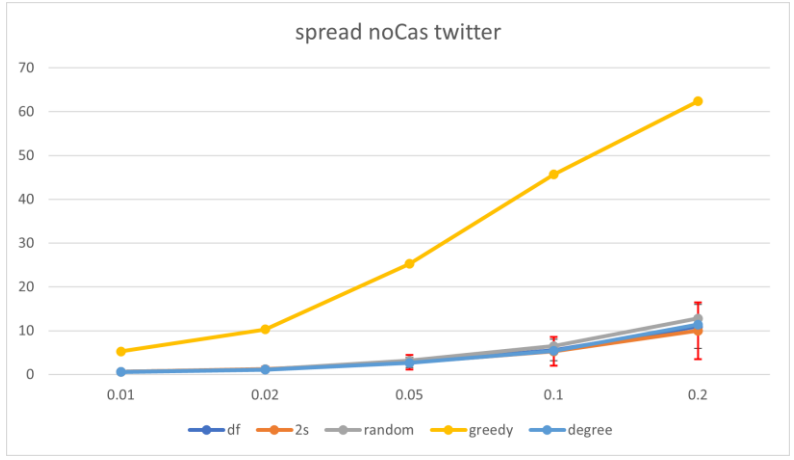


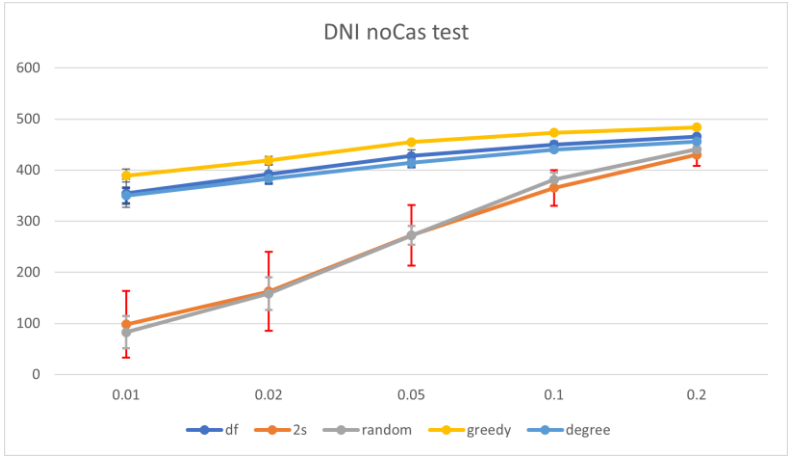


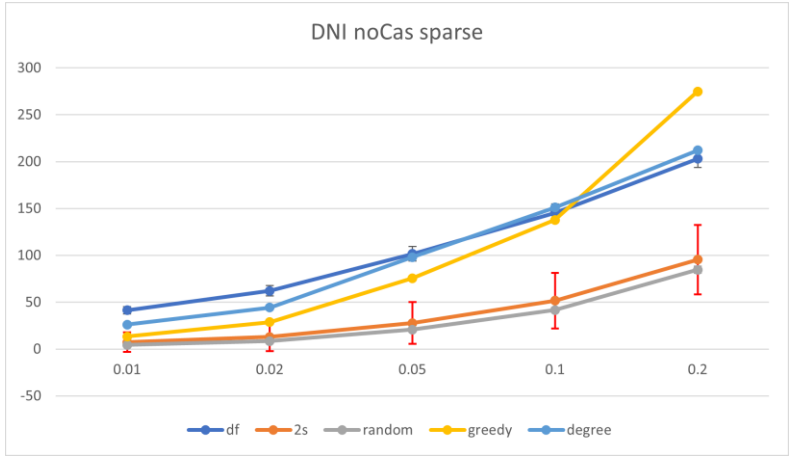
Results : Model 5 : no cascade feature

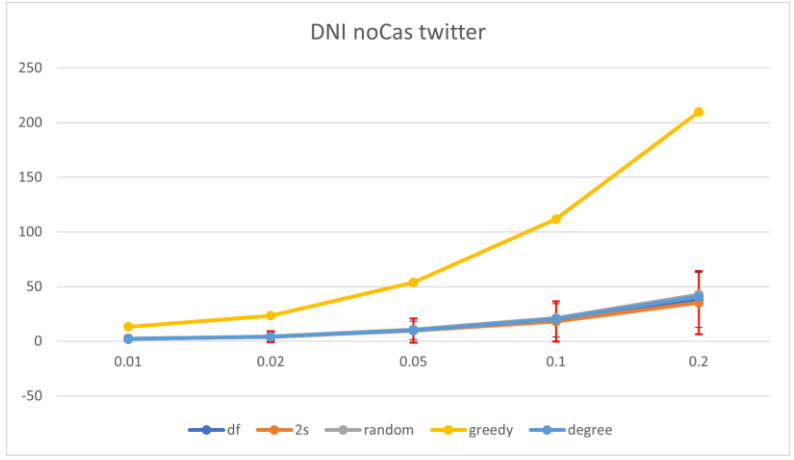












Results : comparison of df models