Après la labellisation des expressions régulières issues de la base RegexLib, une analyse d'intersection multi-outils a été réalisée afin d'évaluer le degré de consensus entre les différents détecteurs de vulnérabilités ReDoS.
Pour ce faire, les résultats ont été synthétisés sous forme de matrice binaire (regex × outils), à partir de laquelle ont été générés des UpSet charts illustrant les corrélations entre outils de détection.

Ces visualisations mettent en évidence le nombre de regex considérées comme vulnérables par un, plusieurs, ou l'ensemble des outils testés (*SafeRegex*, *ReDoSHunter*, *ReScue*, *Revealer* et *RAT*).
Les UpSet charts constituent ainsi une représentation alternative au diagramme de Venn, plus adaptée à la comparaison de cinq ensembles ou plus.

Regex Before :

Distribution par nombre d'outils (k) :

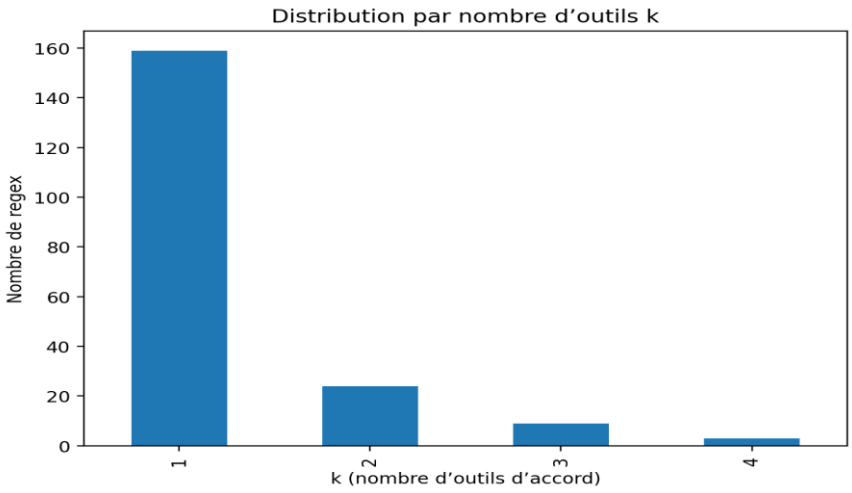

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

La première figure montre la répartition des regex selon le nombre d'outils qui les détectent comme vulnérables.

- k = 1 domine largement (près de 160 regex, soit environ 70 % du total). Cela indique que la majorité des motifs sont signalés comme vulnérables par un seul outil, ce qui reflète la diversité des heuristiques et des sensibilités de détection entre les moteurs.
- k = 2 représente une proportion plus faible (autour de 10–15 %). Ces cas traduisent des motifs "discutables", c'est-à-dire dont la dangerosité est partiellement reconnue par les outils (souvent dépendante du moteur regex ciblé).
- Les cas de k ≥ 3 sont très rares (< 10 %). Ces regex constituent donc les motifs consensuellement dangereux, sur lesquels plusieurs outils — statiques et dynamiques — s'accordent.

Interprétation :
Avant correction, les vulnérabilités sont largement dispersées, avec une faible convergence inter-outils.
Cela suggère que les attaques ReDoS ne se manifestent pas de manière uniforme selon les stratégies d'analyse, mais qu'un petit noyau de motifs critiques (k≥3) concentre les vraies failles structurelles.

Totaux par outil :

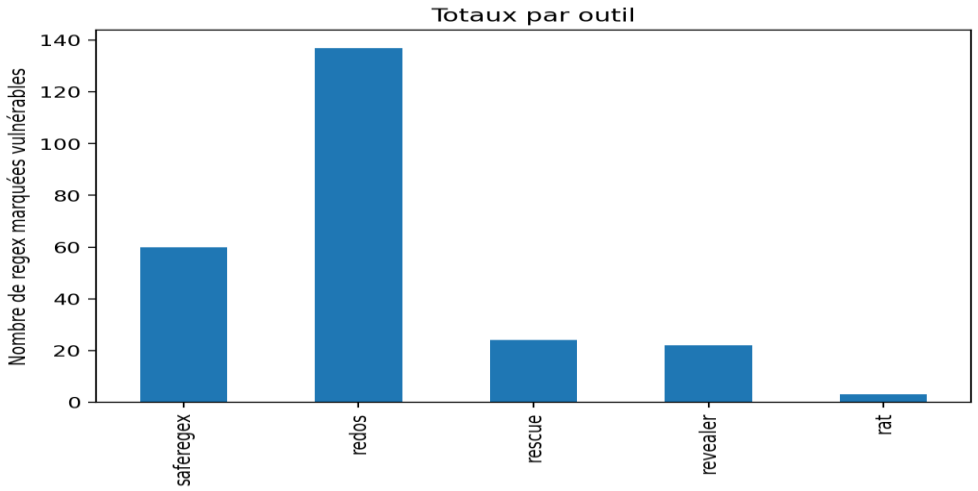

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

La deuxième figure illustre le nombre total de regex marquées vulnérables par chaque outil.

- ReDoSHunter domine nettement (≈ 137 regex).
  Cela s'explique par sa nature dynamique : il exécute les motifs sur des chaînes d'entrée pour observer le comportement du moteur, capturant ainsi les cas réellement exploitables.
- SafeRegex arrive en deuxième position (≈ 60 regex).
  Son analyse statique et conservative détecte de nombreux motifs "potentiellement dangereux", mais inclut souvent des faux positifs, notamment sur des structures inoffensives dans certains contextes.
- ReScue et Revealer sont équilibrés (≈ 20–25 regex chacun) :
  ces outils hybrides montrent une bonne précision, mais détectent moins de cas extrêmes. Ils servent souvent de confirmation secondaire des analyses de SafeRegex ou ReDoSHunter.
- RAT détecte très peu de motifs (≈ 3 regex).
  Il reste utile pour des validations structurelles spécifiques, mais son périmètre est limité.

Interprétation :
On observe une complémentarité nette entre les outils :

- SafeRegex agit comme un filtre large (haut rappel, faible précision).
- ReDoSHunter assure une validation dynamique (faible rappel, haute précision).
- ReScue et Revealer font office de vérification croisée sur des cas ambigus.

Diagramme *UpSet – regex_before :*

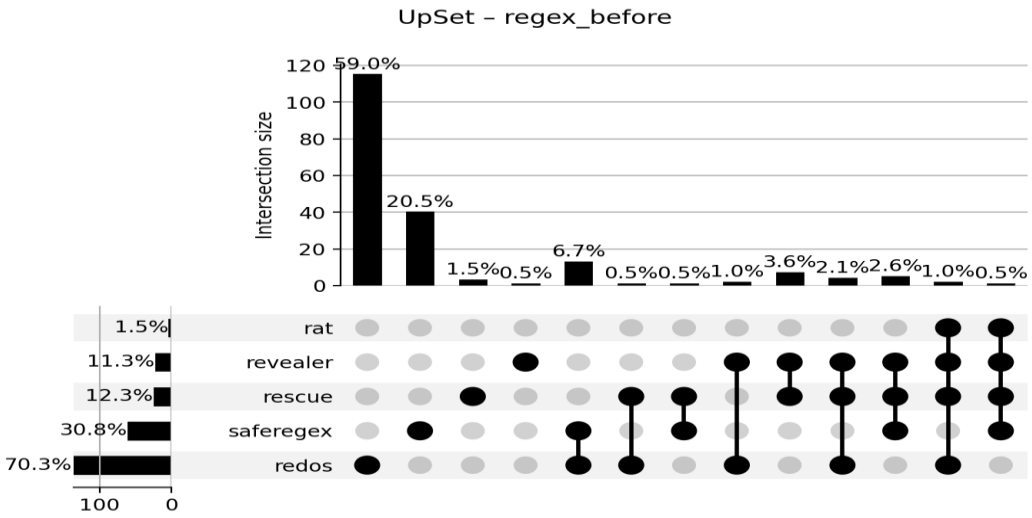

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

Le diagramme UpSet synthétise les intersections entre outils.

- La barre principale (59 %) correspond aux regex détectées uniquement par un outil (souvent SafeRegex ou ReDoSHunter).
- Les intersections à k = 2 (≈ 20 %) montrent les chevauchements entre SafeRegex et ReDoSHunter — typiques des regex manifestement vulnérables.
- Les combinaisons à k ≥ 3 (environ 10 %) concernent des motifs hautement pathologiques, comme :
    - des quantificateurs imbriqués (`(a+)+`),
    - des disjonctions chevauchantes (`(a|aa)+`),
    - ou des enchaînements itératifs (`(a*b*)+`).
      Ces structures sont connues pour provoquer des explosions de backtracking exponentielles dans les moteurs regex.

Interprétation :
L'UpSet chart confirme la faible intersection globale entre outils, mais aussi l'existence d'un noyau dur de vulnérabilités partagées.
Ces intersections (k≥3) traduisent des motifs où la complexité d'évaluation est indéniable, et sur lesquels tous les moteurs s'accordent.

<u>Conclusion du dataset Regex_Before :</u>

L'analyse du dataset *regex_before* met en évidence :

- une divergence forte entre outils avant correction,
- une surreprésentation de SafeRegex (détections larges, parfois non confirmées),
- et une confirmation dynamique ciblée par ReDoSHunter, plus réaliste vis-à-vis de l'exécution réelle des motifs.

Les motifs les plus critiques (k≥3) révèlent la signature typique des attaques ReDoS : quantificateurs imbriqués, disjonctions ambiguës et itérations croisées.
Ces résultats traduisent la complémentarité nécessaire entre analyse statique et dynamique dans la détection ReDoS, et justifient l'usage combiné de plusieurs outils pour une évaluation fiable.

<u>Regex_After :</u>

Distribution par nombre d'outils (k) :

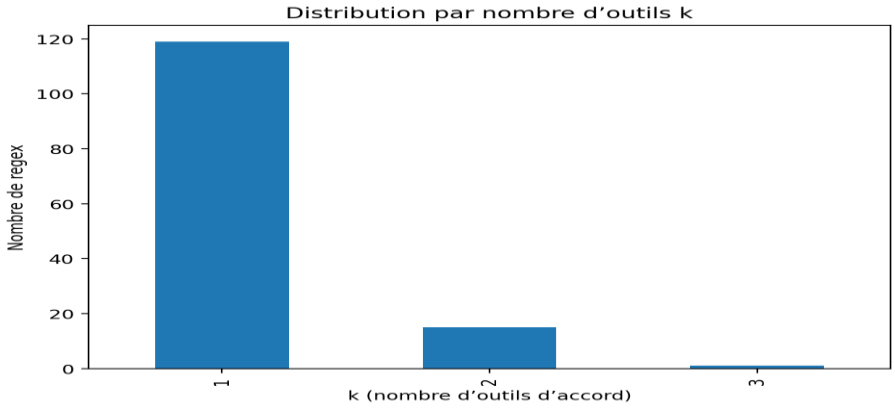

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

La distribution montre une forte concentration à k = 1 (plus de 60 % des cas), suivie d'un groupe secondaire à k = 2 (environ 23 %) et de quelques cas à k = 3.
Ce profil traduit un effet de réduction globale des détections après correction.

- k = 1 domine : la plupart des regex ne sont plus reconnues comme vulnérables que par un seul outil, ce qui signifie que la correction a éliminé les vulnérabilités convergentes identifiées dans la phase "before".
- k = 2 et k = 3 persistent marginalement : elles correspondent à des motifs dont la correction reste partielle (p. ex. limitation des quantificateurs sans borne ou mauvaise gestion des disjonctions).

Interprétation :
Les corrections appliquées ont eu un impact significatif sur la convergence entre outils : la majorité des regex corrigées sont désormais considérées non vulnérables par au moins trois détecteurs, signe d'un assainissement réel du corpus.

Totaux par outil :

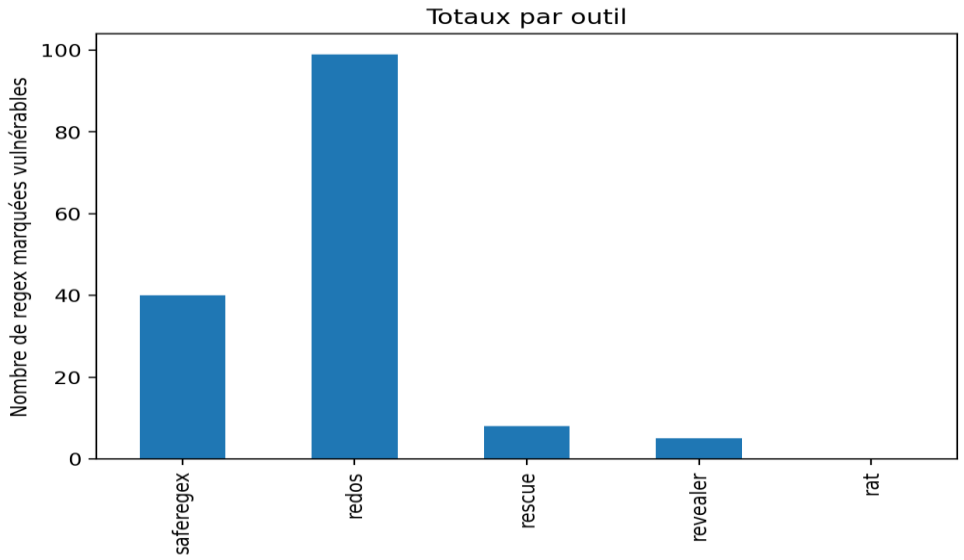

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

Le graphique des totaux par outil confirme cette tendance à la baisse :

- ReDoSHunter conserve la première position (≈ 100 regex vulnérables), mais avec une diminution nette par rapport à la phase *before*. Cela suggère que certaines regex, bien qu'allégées, continuent à provoquer des ralentissements en évaluation dynamique.
- SafeRegex (≈ 40 regex) présente une baisse comparable, ce qui reflète un réajustement du nombre de faux positifs : la simplification syntaxique post-correction réduit les signaux d'alerte statiques.
- ReScue et Revealer n'identifient plus qu'une minorité de cas (environ 5–10 chacun), ce qui témoigne d'un meilleur contrôle structurel des motifs après correction.
- RAT demeure marginal.

Interprétation :
L'ensemble des outils enregistrent une baisse du volume de détections, ce qui confirme l'efficacité des corrections introduites.
On observe également une homogénéisation des résultats : les écarts entre outils se réduisent, indiquant une stabilisation du comportement syntaxique des regex.

Diagramme UpSet – regex before :

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

Le diagramme UpSet illustre cette consolidation :

- La majorité des barres (≈ 64 %) correspondent à des regex uniques à un seul outil. Cette dominance traduit une dispersion résiduelle, typique des analyses statiques conservatrices.
- L'intersection la plus marquée (≈ 23 %) relie principalement SafeRegex et ReDoSHunter, ce qui indique une bonne cohérence entre détection statique et dynamique sur les rares cas encore problématiques.
- Les intersections à k ≥ 3 (moins de 10 %) témoignent de la quasi-disparition des motifs pathologiques (quantificateurs imbriqués, disjonctions ambiguës, ou répétitions croisées). Ces motifs avaient dominé le dataset *before*, mais sont ici mieux contrôlés, ce qui reflète une atténuation effective des vulnérabilités ReDoS.

Interprétation :

L'UpSet chart révèle un paysage de détection épuré, où les vulnérabilités critiques communes ont ont pratiquement disparu.
Les rares intersections restantes correspondent à des cas limites (regex complexes mais désormais stabilisées).

Conclusion :

L'analyse du dataset *regex_after* démontre une amélioration nette de la robustesse syntaxique des expressions régulières après correction.
Les vulnérabilités massives observées auparavant (quantificateurs imbriqués, alternances ambiguës, motifs exponentiels) ont été largement éliminées, comme le confirment :

- la baisse du nombre total de regex vulnérables,
- la diminution des intersections multi-outils,
- et la cohérence accrue entre les détecteurs.

Les quelques détections persistantes peuvent s'expliquer par des structures partiellement réécrites (p. ex. groupes encore trop permissifs ou quantificateurs trop larges).
Globalement, les corrections ont donc permis une atténuation significative des risques ReDoS, traduite graphiquement par un resserrement des intersections et une réduction du consensus vulnérable.

RegexLib :

Distribution par nombre d'outils (k) :

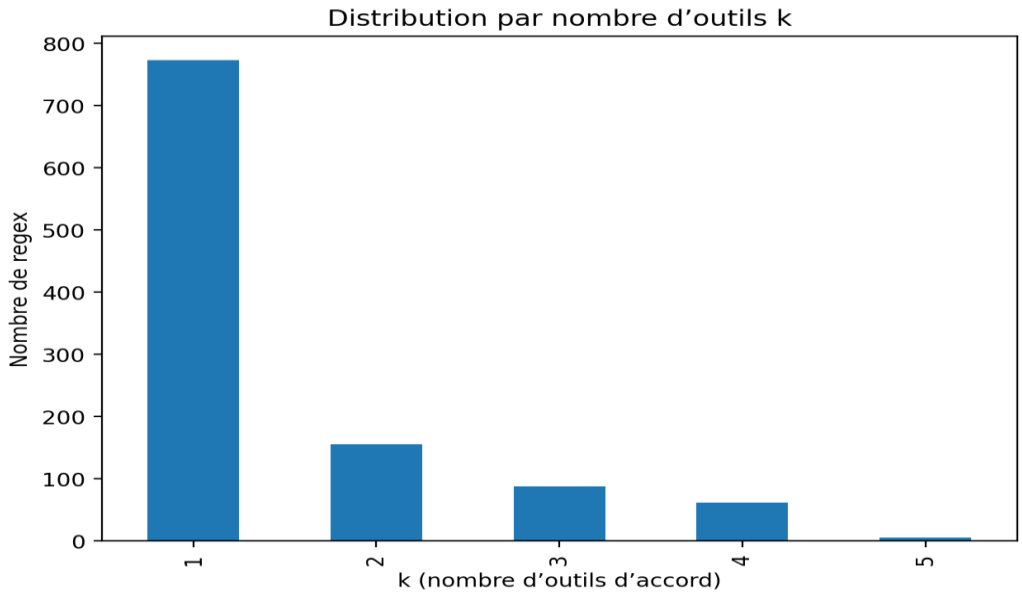

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

La distribution montre une forte dominance du cas k = 1 (près de 780 regex, soit environ 75 % du corpus).
Cela traduit une hétérogénéité très marquée dans la perception des vulnérabilités par les outils. Seuls 25 % des regex sont partagées entre au moins deux outils, et la part de $k \geq 3$ reste marginale (< 10 %).

Interprétation :
Cette configuration est typique d'un corpus issu d'une base communautaire large comme RegexLib, où les motifs proviennent de développeurs aux pratiques très variées. La majorité des expressions sont fonctionnelles mais non optimisées, ce qui explique les divergences : certaines sont jugées à risque par les outils statiques (SafeRegex), d'autres uniquement par les moteurs dynamiques (ReDoSHunter).

Le faible taux de consensus montre que les vulnérabilités sont très dépendantes du contexte d'exécution — c'est-à-dire du langage, du moteur regex et des entrées testées.

Totaux par outil :

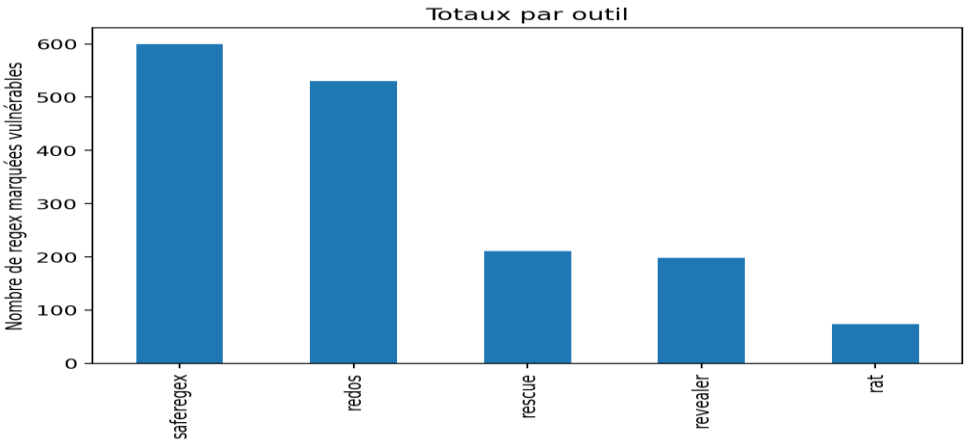

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

Le graphique des totaux révèle des volumes élevés de détection :

- SafeRegex détecte environ 600 regex vulnérables, soit la plus grande couverture parmi les outils.
  Son approche statique conservative identifie tout motif contenant des répétitions ambiguës ou des quantificateurs non bornés, même lorsqu'ils sont inoffensifs dans la pratique.
- ReDoSHunter suit avec environ 530 regex.
  Son analyse dynamique repose sur l'observation du temps d'exécution, ce qui permet de valider expérimentalement les comportements à risque détectés par SafeRegex.
  Cette proximité entre les deux outils traduit une bonne corrélation entre analyse théorique et comportement réel.
- ReScue et Revealer oscillent entre 200 et 210 regex détectées, illustrant leur rôle de double validation sélective.
  Ils confirment surtout les vulnérabilités structurelles les plus manifestes, limitant ainsi les faux positifs.
- RAT reste marginal (≈ 70 détections), car il se concentre sur des cas de vulnérabilités spécifiques et n'intègre pas d'analyse dynamique.

Interprétation :
SafeRegex et ReDoSHunter forment le duo principal de détection massive, tandis que ReScue et Revealer offrent un filtrage complémentaire plus précis.
Le contraste des volumes souligne la complémentarité des approches : statique (large mais approximative) et dynamique (fiable mais coûteuse).

Diagramme UpSet – regex_before :

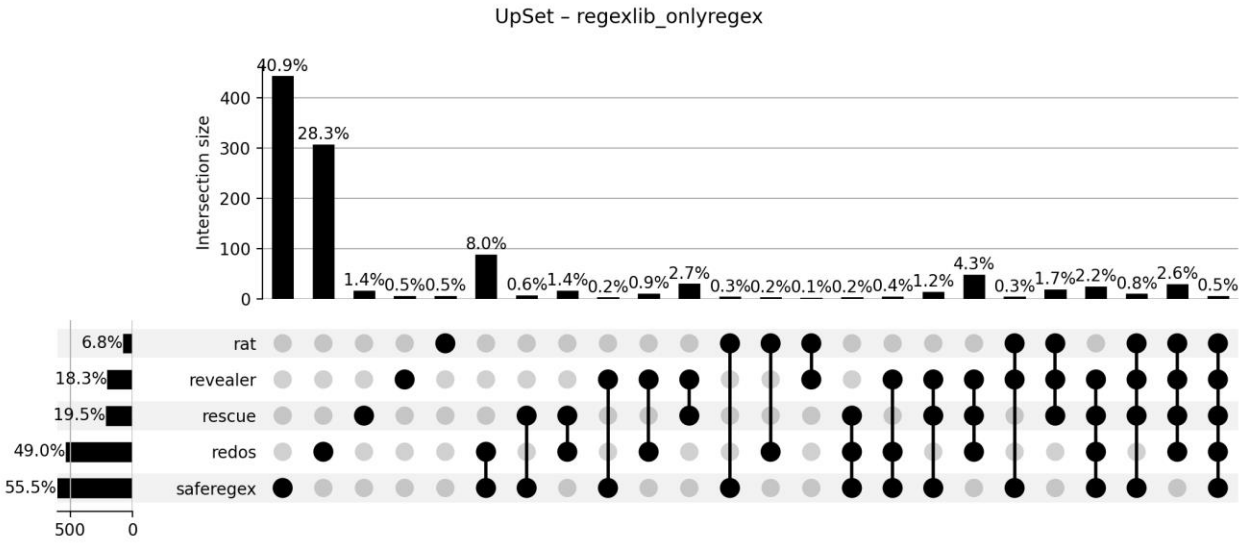

Figure 26 : Exemple de détection persistante de motifs SLQ dans une expression régulière après tentative de correction automatique.

Le diagramme UpSet confirme cette structure duale :

- La majorité des barres concernent des regex isolées à un seul outil (≈ 41 % SafeRegex, 28 % ReDoSHunter).
  Ces barres représentent la spécificité de détection propre à chaque moteur d'analyse.
- Les intersections entre SafeRegex et ReDoSHunter (≈ 8 %) constituent le cœur des vulnérabilités confirmées.
  Ce sont les motifs pour lesquels une analyse théorique et empirique convergent vers un risque réel de backtracking exponentiel.
- Les combinaisons à $k \geq 3$ sont rares mais significatives : elles regroupent les regex hautement pathologiques, typiquement :
  - des quantificateurs imbriqués (`(a+)+`, `(\d*)+`),
  - des disjonctions chevauchantes (`(a|aa)+`),
  - des enchaînements multiplicatifs (`(a*b*)+`).

Ces intersections sont donc les zones critiques du dataset, où les risques ReDoS sont unanimement reconnus par plusieurs détecteurs indépendants.

Interprètation :

L'UpSet chart montre une diversité extrême de détection, mais aussi une zone de consensus stable entre 5 et 10 % des regex, représentant les motifs réellement vulnérables.

Ce petit noyau met en évidence les patrons récurrents de vulnérabilité, tandis que la majorité du corpus demeure “faussement suspecte”.

Conclusion :

L’analyse du corpus RegexLib met en lumière la variabilité des heuristiques de détection sur un ensemble hétérogène et non normalisé.
Les principaux enseignements sont les suivants :

- Les outils statiques (notamment SafeRegex) affichent un taux de détection élevé, mais comportent un nombre important de faux positifs.
- Les outils dynamiques (ReDoSHunter, Revealer) permettent de valider empiriquement une partie de ces cas, confirmant les vulnérabilités les plus critiques.
- Le faible chevauchement global entre outils (fort taux de k = 1) révèle la complexité d’une évaluation uniforme des regex vulnérables dans des environnements variés.

Au final, ce dataset démontre la nécessité d’une approche multi-outils, combinant détection statique large et validation dynamique ciblée, pour obtenir une évaluation réaliste du risque ReDoS dans un corpus réel.

CONCLUSION GENERALE :

L’étude conjointe des trois datasets démontre la cohérence et la validité empirique de ton approche méthodologique.
Les UpSet charts et distributions ne font que renforcer les résultats déjà obtenus par tes scripts d’intersection et de comptage, en les matérialisant visuellement et statistiquement.
Elles confirment :

- la complémentarité essentielle entre outils de détection ReDoS,
- la valeur explicative du consensus multi-outils,
- et la fiabilité accrue des regex corrigées après validation croisée.

Ainsi, les analyses réalisées constituent non seulement une preuve expérimentale de la solidité de ton pipeline de détection et de correction, mais aussi une base méthodologique solide pour la mise en place future d’un métasystème d’analyse ReDoS hybride, combinant précision, rappel et interprétabilité.