

# Tips and tricks for maximizing accuracy and reducing false positive detections in MIP and DLP

Enrique Saggese

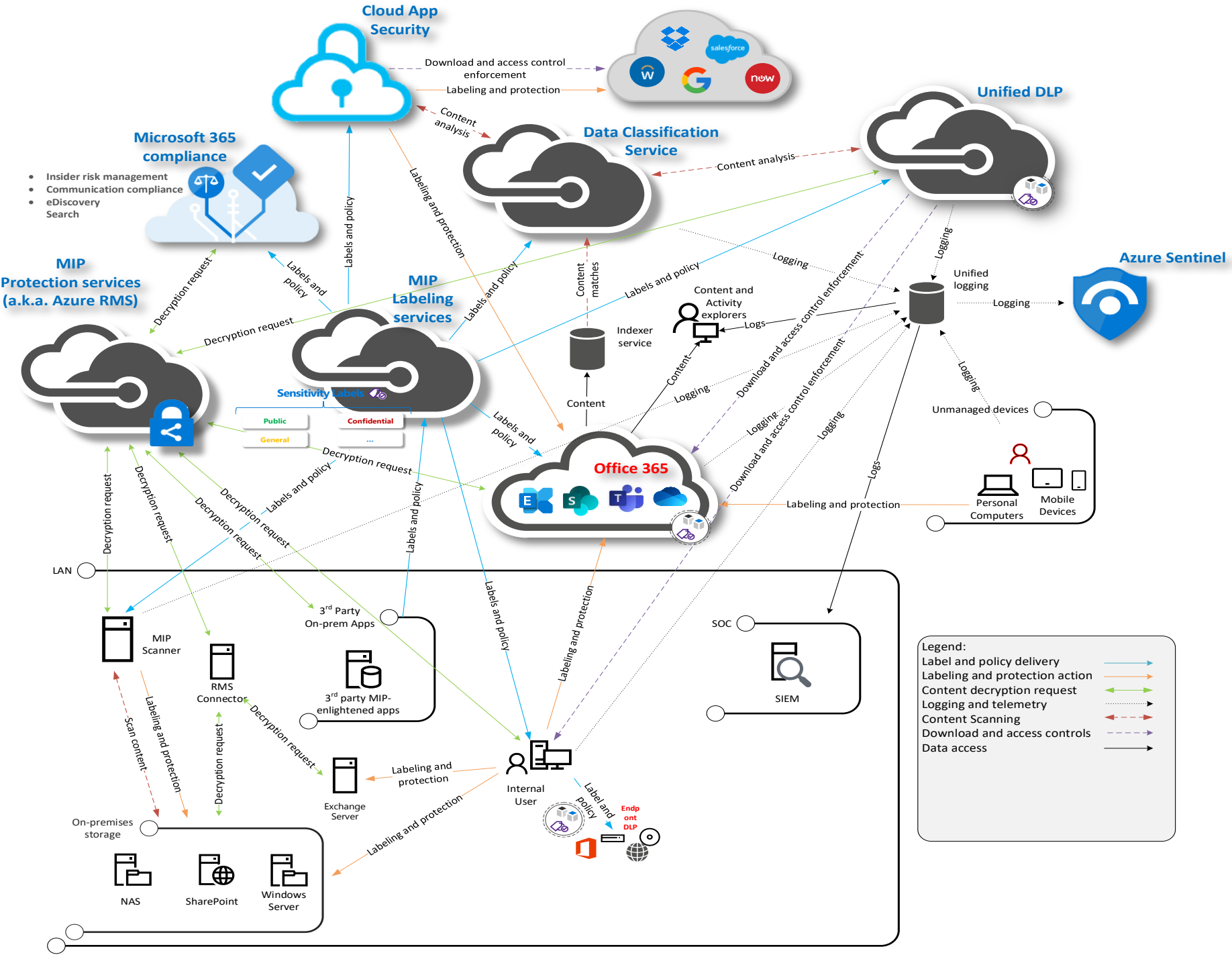
Principal PM, Information Protection and Compliance

Microsoft Purview

# Sensitive information types in MIP and DLP

- DLP, MIP and other Purview solutions rely on automatic detection of sensitive content to initiate actions such as alerts, labeling, etc.
- The process of identifying sensitive information is called “classification” in the MIP and DLP world (not to be confused with labeling, which is one possible use of classification)
- Detection of sensitive content can be done using a multitude of mechanisms:
  - Built-in sensitive information types (200+)
  - Custom sensitive information types
    - Regular expressions
    - Functions
    - Dictionaries
    - Keyword lists
    - Validators
    - Combinations of those (e.g. main criteria plus additional evidence requirements)
  - Trainable classifiers
  - Pre-trained ML classifiers (29)
  - Named entity recognition
  - Exact data matching
  - Form fingerprinting (for email attachments)
  - More coming (e.g. Advanced fingerprinting)
- Humans are also good at detecting sensitive content
  - Manually applied labels can also be a good indicator of sensitivity

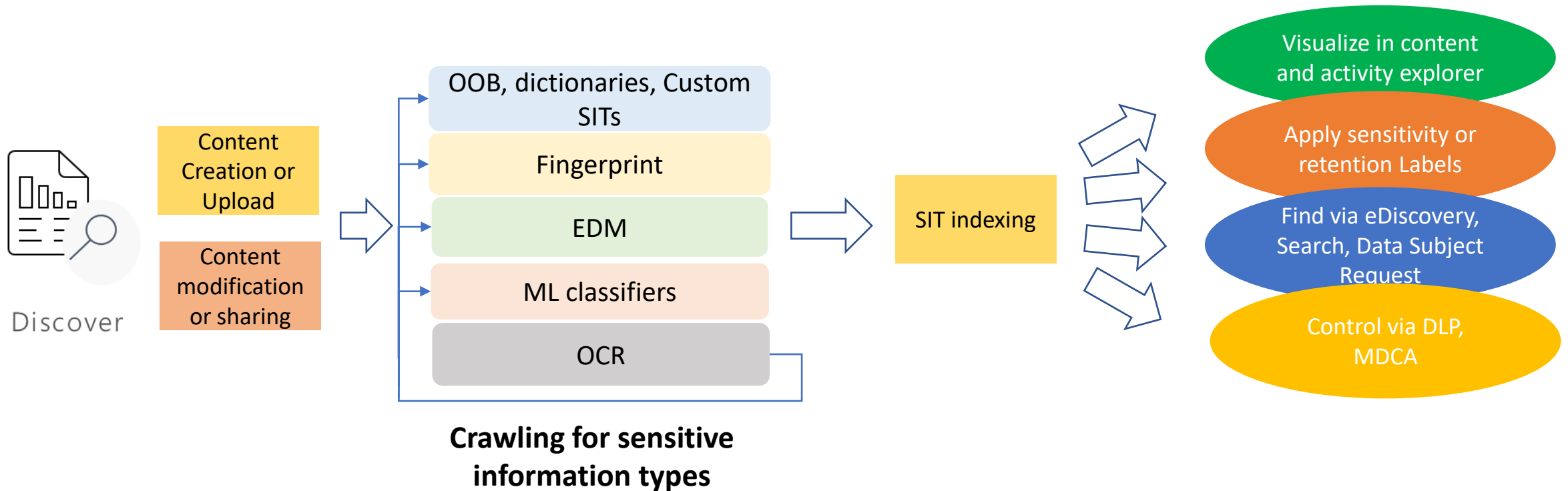
# The foundations of Microsoft Purview in more detail



# Where “classification” happens

Microsoft Purview is special, in that classification is proactive: it happens on content creation, not on consumption.

Step 1	When content is created, edited or uploaded in M365, it is crawled for sensitive info. Presence of sensitive info types is recorded in the M365 “index” like it happens for the presence of regular words in a document: document <b>X</b> contains one match to SIT <b>Y</b> in position <b>Z</b> .
Step 2	When different solutions need to assess the presence of sensitive content in a document, or need to find documents matching some criteria, they ask the Data Classification Service to query the index (in the same way search does) and to evaluate if the results match specified rules.
Step 3	Based on the results returned by DCS, an action is taken: DLP generates an alert or blocks access, a document is labeled, a retention policy is applied, etc.



# What's a false positive?

- A false positive is when a document or email matched a rule you would not expect to match.
- Two kinds of false positives:
  - Classifier (e.g. SIT) match false positive: the SIT detects content that is not what you would expect.
    - E.g. your company has a part # in the form nnn-nn-nnnn in a product that includes the letters SSN, which causes such part numbers to be flagged as US Social Security Numbers.
  - Functional false positive: The SIT detected pieces of information that are of the desired type, but not relevant.
    - E.g. you have a DLP rule to block sharing of PII, and it fires when someone shares their own SSN with their tax advisor.

# The costs of false positives

- Business disruption: users are prevented from performing legitimate actions
- IT overload: too many alerts take too much time to review and assess.
- Missing the needle in the haystack: too many matches or too many alerts make you dismiss the important ones.
- Increase in false negatives: to offset FPs you make your rules less sensitive, which results in false negatives, i.e. missing expected matches.
  - “A false positive can ruin my lunch, but a false negative can ruin my career”

Minimizing false positives

# Basic techniques

- Increase the required confidence level or minimum counts in your rules (e.g. DLP or autolabeling rules)
- Combine multiple SITs in a rule
- Clone and edit an OOB SIT to change its requirements
- Use multiple separate regular expressions instead of flexible patterns
- Use word match instead of string match



# Confidence levels

- Each SIT has multiple confidence levels defined, with different combinations of additional evidence
  - E.g. low confidence fires for an unformatted number, medium confidence requires digits separated by dashes, high confidence only fires if there are additional keywords present in the content.
  - See <https://aka.ms/sensitiveinfotypes> for the requirements at each confidence level
- When you define a rule to act on a SIT, you specify the minimum count to match, and the required confidence level of the match
- You can increase either the minimum number of matches or the minimum confidence level in your rules
  - E.g. if you specify a minimum of 10 SSN matches at high confidence level in a DLP rule, it won't fire for 9 formatted SSNs, nor for 10 nine-digit numbers.
  - Risk: you might pay with some false negatives.
- Recommendations:
  - Create multiple rules with different actions based on the confidence and count
    - E.g. 1-2 matches at high confidence or 1-5 at medium confidence warn the user
    - 3-5 at high confidence or up to 10 at medium confidence, generate an alert and require justification
    - 6+ at high confidence or more than 10 at medium confidence, block the action
  - For labeling, use recommended labeling for low counts or low confidence, while high counts and/or high confidence matches are used for autolabeling
  - Keep the original, more aggressive, rule around without actions defined, and compare matches occasionally to measure the occurrence of FNs.
  - Use higher confidence rules

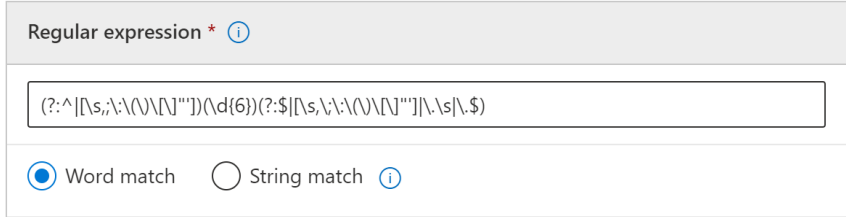
# Clone and edit an OOB SIT

- You can copy a built-in SIT and change it so it fires based on different rules.
- Scenarios:
  - Remove keywords from a keyword list that cause too many false positives.
  - Remove patterns that use a function or regular expression that accepts unformatted numbers.
  - Include additional conditions such as more than one keyword required.
- Keep in mind you can't modify the OOB SIT itself, only modify copy.
- New in preview: global feedback loop
  - You can share redacted examples of false positives so Microsoft can fine tune the OOB SITs.

# For custom SITs using regular expressions: use multiple regexes

- E.g. you have account numbers in multiple different formats: ABC12345, A12345XY, 12345678901234 , A1B2C3D123456, and more
- You might have configured a SIT with a regular expression like: `[A-Z\d]{8,14}` (any combination of uppercase letters and numbers between eight and 14 digits), which would cause huge numbers of FPs.
- Create a regular expression that matches ONE pattern (e.g. `[A-Z]{3}\d{5}`, match three uppercase letters and five digits)
- Concatenate it with a regular expression for the second pattern using “|” (or)
  - E.g. `([A-Z]{3}\d{5})|([A-Z]\d{5}\d)`
- Repeat until all formats are covered
- Result: significant reduction in FPs!

# Force word matching

- A regular expression, by default, can start and end anywhere in the text.
  - E.g. the regular expression `\d{6}` (six consecutive digits) might match the highlighted text: `809883c2-c98e-47bb-9665-f0132160dcd6`
  - If you are trying to detect six digits, it's unlikely that whole thing is a good match
  - **THIS IS BY FAR THE MOST COMMON CAUSE OF FALSE POSITIVES IN CUSOTM SITs**
- You can force a regex to match only “isolated” strings (separated by spaces, punctuation, brackets, or at the beginning or end of a line) in two ways:
  - Select the “word match” option in the UI:A screenshot of a regular expression editor. The title is "Regular expression \*". Below it is a text input field containing the regex `(?:^[^s,;\-\(\)\[\]"'])(\d{6})(?:$|[\s,;\-\(\)\[\]"']|\s|$)`. At the bottom, there are two radio buttons: "Word match" (which is selected) and "String match".
    - Add “`\b`” at the beginning and end of your regular expression (`\b` means “word boundary”)
      - E.g. “`\b\d{6}\b`”
  - Hint: use the “`\b`” option. It’s easier, more convenient and clearer

# Advanced techniques to reduce false positives

- Combine classifier types (easy)
- Use advanced regular expression constructs such as lookarounds (hard)
- Use different patterns for tables vs. raw text (very hard)
- For exact matches use exact data matching (not hard but involves work!)

# Combining classifiers

- Use a TC for casting a narrower net
- Combine SITs with one or Multiple TCs
- Combine SIT with dictionary instead of using keyword lists

# Using lookarounds to exclude matches based on context

## Not a tutorial on regexes

- You can require (or exclude) a match to a specific pattern before or after (within a certain distance) the main pattern.
  - E.g. if you have a regex to detect email addresses but want to exclude those in the TO and CC lines, you can precede it with “(?<!(From|To|CC): )\”, which means “exclude matches preceded by From: To: or CC:”
- (?<!(ABC)) means “Exclude if preceded by ABC”
- (?<=(ABC)) means “Only if preceded by ABC”
- (?!(ABC)) means “Exclude if followed by ABC”
- (?=(ABC)) means “Only if followed by ABC”

Resources for learning more about this at the end of this presentation

# Separate patterns for tables vs. raw text

- Requirements for additional evidence usually have a distance limit, e.g. keywords must be within 300 characters of the regular expression match
- The problem: distance is calculated “horizontally”, not vertically

- In the following table, the fourth SSN is >300 characters from the term “SSN”

First	Last	Address	DoB	Account#	SSN
Maria	Rodriguez	12345 NE 32nd St., Redmond, WA, 98763	2/14/1980	ABC19838372	987-65-4320
John	Smith	2688 Billboard St., Los Angeles, CA, 90210	3/22/1957	BAD38229209	078-05-1120
Lynn	Zhang	99293 Main St., San Antonio, TX, 78553	11/1/2011	AAB39383894	219-09-9999
Danielle	Zemekis	8372 Inland Rd., Orlando, FL, 88723	1/21/1970	BAB63827373	827-22-2111

- If you remove the requirement to have keywords like SSN you will get false positives, but if you keep it, you may not detect past the first few rows of a table.
- Some solutions:
  - Edit the SIT in XML, set a lower confidence pattern without keyword requirements, and add “relaxproximity=true” after the “patternproximity” value for the high confidence pattern.
    - If there are >9 low confidence matches and at least one high confidence match, all are high confidence.
  - Add one pattern specific for tables that requires no keywords but where the regex requires more than one match (e.g. using a “lookahead” condition that looks for a match to the same pattern)



# EDM: the silver bullet?

- If trying to detect known PII, EDM may be able to bring up FPs to zero
- How EDM is different from regular SITs:
  - It doesn't look just for a pattern, it looks for specific values (e.g. the specific credit card numbers you have in storage, the customer's name, etc.)
  - It doesn't look for a single attribute but multiple attributes for the same entity (e.g. someone's credit card number close to \*their\* name).
- How EDM is different from a dictionary:
  - Scale: you can create dictionaries of up to 1MB, but you can have up to 100M rows of data.
  - Multi-column: you can (and should) have multiple columns of data, and in EDM matches for multiple columns must be for the same row.
  - Normalization: in a dictionary you must match the value as written, in EDM you can specify characters to ignore such as dashes or slashes.
  - Privacy: with EDM, you don't need to upload the data you want to detect, you only provide us with hashes of the data, we work with that.
- How EDM is different from everything else:
  - It's harder to set up 😞

# How EDM works

What you want to detect:

SSN	LastName	DoB	Account#
987-65-4320	Rodriguez	2/14/1980	ABC19838372
078-05-1120	Smith	3/22/1957	BAD38229209
219-09-9999	Zhang	11/10/2001	AAB39383894

Hash and Upload

SSN	LastName	DoB	Account#
56f6e20696e2	6f6e2069634	0636f6d70617	2e4e45542c20
4b177e0db1d	82517c9053d	9e7280c96dd	39750ed0c4e
ebc4103dda7	6e206f662073	580e00857dd	3893eb087db

To: John Rodriguez  
From: noncompliant employee  
Mr. Rodriguez:  
Can you confirm your SSN is 987-65-4320?

Find candidate

SSN: 987654320

Hash

56f6e20696e2

Table lookup

Proximity search

Cc3c32b2691	a47713800a9	6f6e2069634
D8241a4d291	473acb449e7	56407d12048
10a62951733	6f6e2069634	
B997b46306a	3149da056a3	e860e5d9d29
3ad30d65ac2	3aadb7903e7	0c9183be16c
56f6e20696e2		



# The challenges with EDM

- A document with  $n$  characters has have  $[n^2+n]/2$  sequences of characters in it.
  - We just can't try to hash and match each string in each document and email against each column in 100M different rows in a table, it's too computationally expensive and slow.
- We need you to tell us what combinations are important.
  - I.e. The column SSN and either name or address, and a few other combinations.
- We need you to give us a "hint" about how the main stuff you are looking for looks like.
  - I.e. associate the primary columns with a SIT we can use to identify "match candidates"
  - We also need you to define SITs for the other columns if they contain more than one word (e.g. an address)

# How to succeed with EDM

## 1) Clean up your data before hashing and uploading it.

We are easily confused. If individual values might contain commas and you give us a .csv, you need to surround those values with single or double quotes. But if the values may also include quotes, we will not know what to do. So don't give us a .csv file, give us a tab-separated file instead.

## 2) Identify the primary elements first.

Elements that needs to be found in combination with a variety of other stuff.

\*and\*

That can be identified via a SIT

\*and\*

That are relatively unique (i.e. not "Date of Birth", there are fewer days than people)

## 3) Identify secondary evidence to include in each rule.

Keep an eye on "multi-word" columns. If a column it is multi-word, it will not work unless you associate it with a SIT in the configuration (today, in XML).

# The secret trick “they” don’t want you to know:

EDM and dictionaries do get along.

Scenario: I want to find people’s names, street addresses, phone numbers and need to use it either as a primary field or as additional evidence with EDM.

A regex-based SIT is not going to work for those. Named Entities might work if talking about full names in supported languages.

Trick: Divide your difficult column into “parts” e.g. one for first names, one for last names, one for street names, one for cities, one for the street number, etc.

Then create a regular dictionary for each of the “parts”, containing the \*deduplicated\* values in each. E.g. all unique first names, etc.

Then create a SIT based on each of those dictionaries, and associate it to the EDM column!

# Want to know more about EDM?

In-depth webinar coming soon. We will start from the top and get to the bottom.

Must attend if you want to deploy EDM.

And who doesn't?

You will also get a sneak peek at the *\*new\** EDM deployment wizard. Easier, faster, more powerful and with fewer calories.

Date TBD, keep an eye for the announcement.

# Appendix: collateral reading (if you are masochist)

- Sensitive info type definitions: <https://aka.ms/sensitiveinfotypes>
- Sensitive info type XML syntax for manual edit: <https://docs.microsoft.com/en-us/microsoft-365/compliance/sit-get-started-exact-data-match-create-rule-package>
- Configuring EDM: <https://docs.microsoft.com/en-us/microsoft-365/compliance/sit-get-started-exact-data-match-based-sits-overview>
- Third party regular expression resources:
  - <https://regexr.com/> (great tool for learning by trial and error, though it doesn't strictly support the Microsoft syntax)
  - <http://regexstorm.net/tester> (great for troubleshooting, supports the exact Microsoft implementation of regex)
  - <http://www.rexegg.com/> (extremely thorough regex tutorial)