# TF–IDF: Explanation, formulas, and how to interpret your TF-IDF matrix

> A single Jupyter notebook **Markdown** cell that explains TF-IDF (raw and normalized), how sklearn computes it (defaults), the role of normalization, and a short interpretation of your example TF-IDF output.

---

## 1. Quick overview

**TF–IDF** (Term Frequency — Inverse Document Frequency) is a weighting scheme that measures how important a word is to a document in a corpus. It increases with the number of times a word appears in the document (term frequency) and decreases with how common the word is across the corpus (inverse document frequency).

Why use TF–IDF? - Reduce the impact of common words (e.g. *and*, *the*) that do not carry distinguishing information. - Highlight words that are relatively frequent in one document but rare across others — those are often the most *informative* for that document.

---

## 2. Core formulas (sklearn defaults)

Let: - $tf_{d,t}$ = raw term frequency of term $t$ in document $d$ (i.e., count). - $n$ = total number of documents in the corpus. - $df_t$ = number of documents that contain term $t$ .

**IDF with smoothing (sklearn's default `smooth_idf=True`):**

$$\mathrm{idf}(t) = \log\left(\frac{1+n}{1+df_t}\right) + 1$$

This avoids division-by-zero and pushes idf into a positive range (minimum 1).

**Raw TF–IDF (before normalization):**

$$\mathrm{tfidf}_{d,t}^{raw} = tf_{d,t} \times \mathrm{idf}(t)$$

**Sublinear TF (optional):** if `sublinear_tf=True` sklearn uses

$$\mathrm{tf}_{d,t}^{sublinear} = 1 + \log(tf_{d,t})$$

**Normalization:** after computing the raw TF–IDF vector for a document $\mathbf{v}_d = [\text{tfidf}_{d,1}^{raw}, \text{tfidf}_{d,2}^{raw}, \dots]$ sklearn can normalize using either L2 (default), L1, or None.

- **L2 normalization** (default):

$$\mathbf{v}_d^{(norm)} = \frac{\mathbf{v}_d}{\|\mathbf{v}_d\|_2}, \quad \|\mathbf{v}_d\|_2 = \sqrt{\sum_i v_{d,i}^2}$$

- **L1 normalization**:

$$\mathbf{v}_d^{(norm1)} = \frac{\mathbf{v}_d}{\|\mathbf{v}_d\|_1}, \quad \|\mathbf{v}_d\|_1 = \sum_i |v_{d,i}|$$

- **No normalization**: keep raw TF–IDF.

**Why normalize?** - Normalization prevents long documents from dominating similarity metrics by length alone. L2 normalization makes every document vector length 1 (unit vector), which means cosine similarity becomes the dot product of normalized vectors.

---

## 3. sklearn parameters to remember

- `use_idf=True` (default): apply IDF. If `False`, only TF (optionally sublinear) is used.
- `smooth_idf=True` (default): adds 1 to numerator and denominator in the idf formula to smooth extremes and avoid division-by-zero.
- `sublinear_tf=False` (default): if set to `True`, replace raw counts with `1 + log(tf)` to dampen very frequent terms.
- `norm='l2'` (default): L2 normalization. Options: `'l2'`, `'l1'`, or `None`.

**Typical pipeline (sklearn):**

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
cv = CountVectorizer(lowercase=True)
X_counts = cv.fit_transform(docs)
transformer = TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True,
sublinear_tf=False)
X_tfidf = transformer.fit_transform(X_counts)
```

---

## 4. Interpreting the TF–IDF matrix you provided

You provided a TF–IDF matrix (6 documents × vocabulary size ≈ 61) and a matching `Index([...])` listing the vocabulary terms in order. A few practical points for reading this:

- Each **row** corresponds to a document and is the normalized TF–IDF vector for that document (after L2 scaling).

- Each **column** corresponds to one vocabulary term; that same column index maps to the `Index([...])` list you printed.
- A value near **0** means the term is either absent or extremely uninformative for that document.
- Larger positive values mean the term is relatively important to that particular document compared to other documents in the corpus.

**Example — the token `'and'`:** - In your earlier manual computation `'and'` had a normalized TF–IDF ≈ `0.1948` for document 1. In the matrix you pasted, the column for `'and'` (look up `'and'` in the `Index`) has a value `0.28280113` in the first row. Differences like this can come from different tokenization or whether stop words were removed or how punctuation/word boundaries were detected. But the interpretation is the same: - `'and'` appears in multiple documents → higher document frequency (df) → **lower IDF** → relatively low TF–IDF. - Stop words are frequent across documents and therefore have **small weights** after idf down-weighting + normalization.

**Example — a high TF–IDF word:** - Look for columns with a relatively large value in **one row** and very small values elsewhere. That signals the word is rare in the corpus but used in that document — a strong contextual marker. - For instance, if you see a value `0.31584299` for the term `'education'` only in row 4, that suggests `'education'` is a distinguishing word for document 4.

---

## 5. Notes & best practices for recording and remembering

- ✅**Remember the two parts**: TF (how common in the document) × IDF (how rare across documents). IDF downweights common words.
- ✅**Remember smoothing**: `smooth_idf=True` gives $\text{idf} = \log((1+n)/(1+df)) + 1$ . This shifts idf upward and avoids zeros.
- ✅**Normalization**: L2 scales each document vector to unit length so documents are comparable irrespective of length.
- ✅**Sublinear TF**: helps when you want to dampen the effect of very high term counts (e.g., repeated terms artificially boosting importance).
- Consider **removing stop words** (English stop list) with `CountVectorizer(stop_words='english')` if you want the pipeline to ignore very common function words entirely. That usually reduces noise.

---

## 6. Short interpretation statement you can paste into your notes

- A **low TF–IDF** score (near 0) for a token like `and` means the token appears in many documents (high `df`) so its IDF is low; it's a stop-word-like token and adds little to document identity.
- A **high TF–IDF** score (larger value concentrated in a single document) indicates a token that is rare across the corpus but frequent in that document — a likely **keyword** that helps identify the document's topic.

---

## 7. Quick checklist for reproducibility (copy into your notebook)

- Save the exact `CountVectorizer` settings: `lowercase`, `token_pattern`, `stop_words`.
- Record `TfidfTransformer` settings: `norm`, `use_idf`, `smooth_idf`, `sublinear_tf`.
- Save the `vocabulary_` ordering (so you can map column index → token).
- If you want deterministic results, also record any custom pre-processing (stemming, lemmatization) and the Python / sklearn version.

---

### Conclusion (one-sentence):

TF–IDF highlights the relative importance of words by combining per-document frequency with corpus-level rarity; normalization (L2 by default) makes document vectors comparable so that common words like `and` receive low scores and rare, topic-specific words receive high scores.

*End of notebook markdown cell.*