

This is a copy of a chat between Claude and Andreas. Content may include unverified or unsafe content that do not represent the views of Anthropic. Shared snapshot may contain attachments and data not displayed here. [Report](#)

I have collected a dataset of news articles from the past few weeks. I would like to know which words occur significantly more or less frequently in the last week compared to the four weeks before. What are good methods to do this?

There are several good methods for detecting statistically significant shifts in word frequency between two time periods. Here's an overview:

Corpus Comparison Methods

1. Log-Likelihood Ratio (G^2)

The most widely used method in corpus linguistics (Dunning, 1993). It compares observed vs. expected frequencies under the null hypothesis that both corpora are drawn from the same distribution. It handles low-frequency words better than chi-squared and works well even when the corpora are very different sizes — which matters here since you're comparing 1 week vs. 4 weeks.

2. Chi-Squared (χ^2) Test

A simpler alternative, but it's unreliable for low-frequency words (expected count < 5) and can produce many false positives in sparse data. Log-likelihood is generally preferred over it for text.

3. Mann-Whitney U / Permutation Tests

Non-parametric tests that make no distributional assumptions. Useful if you want to compare word rates *per article* rather than pooled counts, since they respect article-level variance.

Effect Size Measures (use alongside significance)

Significance alone doesn't tell you how *much* a word has changed. Pair it with:

- **%DIFF** — simple percentage difference in relative frequency, easy to interpret
- **Bayes Factor (BIC approximation)** — penalises complexity, good for prioritising the most meaningful differences
- **Log Ratio** — $\log_2(\text{freq_week5} / \text{freq_weeks1-4})$, gives a symmetric, interpretable effect size: +1 means doubled, -1 means halved

Practical Considerations

Issue	Recommendation
Unequal corpus sizes	Normalize to relative frequencies (per million words) before comparing
Multiple comparisons	Apply Benjamini-Hochberg FDR correction — you're testing thousands of words simultaneously
Low-frequency words	Set a minimum frequency threshold (e.g., appears ≥ 5 times total) to reduce noise
Stop words	Decide whether to include them — they shift too and can be informative (e.g., rising use of "we" vs. "they")
Multiword expressions	Consider bigrams/trigrams if single words feel too decontextualised

Recommended Pipeline

1. **Tokenize and normalize** (lowercase, optional lemmatization)
2. **Count frequencies** in each time window
3. **Filter** by minimum frequency
4. **Run Log-Likelihood** for significance; compute **Log Ratio** for effect size
5. **Apply FDR correction** across all words
6. **Rank** by effect size among significant words

Tools & Libraries

- **Python:** `scipy.stats` for chi-squared/ G^2 , `statsmodels` for FDR correction, `nltk`/`spacy` for tokenization
- **R:** the `quanteda` package has `textstat_keyness()` built-in, which implements log-likelihood and several other keyness measures out of the box — very convenient for exactly this use case
- **Ready-made:** AntConc (free GUI tool) supports keyness analysis directly

Would you like help implementing any of these in code?

For the %DIFF, do you calculate it as "freq_week5 - freq_weeks1-4" ?

Can you show scipy.stats code for chi-squared and G^2 ?

%DIFF Formula

It's the relative difference in *normalised* frequencies (not raw counts, since your corpora are different sizes):

$$\%DIFF = ((freq_B / size_B) - (freq_A / size_A)) / (freq_A / size_A) * 100$$

Where A is the reference corpus (weeks 1-4) and B is the target (week 5). So it answers: "by what percentage did the relative frequency go up or down?"

Chi-Squared and G^2 in scipy

```
python

import numpy as np
from scipy.stats import chi2_contingency, chi2

def corpus_stats(word, counts_recent, counts_reference, total_recent, total_refe
    """
    word            : the word being tested
    counts_recent    : raw count in week 5
    counts_reference : raw count in weeks 1-4
    total_recent     : total tokens in week 5
    total_reference  : total tokens in weeks 1-4
    """
    a = counts_recent      # word in recent
    b = counts_reference    # word in reference
    c = total_recent - a    # non-word in recent
    d = total_reference - b # non-word in reference

    contingency = np.array([[a, b],
                             [c, d]])

    # --- Chi-Squared ---
    chi2_stat, p_chi2, _, _ = chi2_contingency(contingency, correction=False)

    # --- Log-Likelihood ( $G^2$ ) ---
    #  $G^2 = 2 * \sum(\text{observed} * \log(\text{observed} / \text{expected}))$ 
    # scipy's chi2_contingency with lambda_="log-likelihood" computes this
    g2_stat, p_g2, _, _ = chi2_contingency(contingency, lambda_="log-likelihood")
```

```

# --- Effect sizes ---
freq_recent      = a / total_recent
freq_reference   = b / total_reference

pct_diff = (freq_recent - freq_reference) / freq_reference * 100

# Avoid log(0) with a small epsilon
eps = 1e-9
log_ratio = np.log2((freq_recent + eps) / (freq_reference + eps))

return {
    "word":          word,
    "freq_recent":   freq_recent,
    "freq_reference": freq_reference,
    "pct_diff":      pct_diff,
    "log_ratio":     log_ratio,
    "chi2":          chi2_stat,
    "p_chi2":        p_chi2,
    "g2":            g2_stat,
    "p_g2":          p_g2,
}

```

A few notes:

- `correction=False` disables Yates' continuity correction for chi-squared, which is standard for large corpora
- `lambda_="log-likelihood"` is scipy's built-in way to get G^2 ; it uses the same contingency table logic
- Both tests have **1 degree of freedom** for a 2×2 table, so the p-values are directly comparable

Running it over a full vocabulary

```

python

from statsmodels.stats.multitest import multipletests

results = [
    corpus_stats(word, counts_recent[word], counts_reference.get(word, 0),
                  total_recent, total_reference)
    for word in counts_recent
]

```

```
# FDR correction across all words
p_values = [r["p_g2"] for r in results]
_, p_adjusted, _, _ = multipletests(p_values, method="fdr_bh")

for r, p_adj in zip(results, p_adjusted):
    r["p_g2_adjusted"] = p_adj
```

Would you like help loading your articles and building the frequency counts to feed into this?

And what about tf-idf?

TF-IDF is a slightly different tool — worth understanding the distinction before deciding whether to use it.

What TF-IDF actually measures

TF-IDF scores a word relative to how *distinctive* it is within a document compared to a collection. The formula is:

$$\text{TF-IDF}(\text{word}, \text{doc}) = \text{TF}(\text{word}, \text{doc}) \times \text{IDF}(\text{word}, \text{corpus})$$

- **TF** — how often the word appears in a specific document
- **IDF** — penalises words that appear in many documents (so "the" scores low)

It was designed for **ranking documents by relevance to a query**, not for comparing two time periods.

The core problem for your use case

If you treat each time period as one big document, TF-IDF collapses to just IDF — because TF becomes the same for both "documents" and cancels out. You'd just be finding words that are rare across the whole corpus, which isn't what you want.

You *could* treat each article as a document and then aggregate scores by time period, but then you're measuring **within-article distinctiveness**, not **frequency shifts over time**.

When TF-IDF is useful in your pipeline

Use case	Good fit?
Detecting words that increased/decreased in frequency week-over-week	✗ Use G^2 + log ratio instead
Finding the most topically distinctive words <i>within</i> a single article	✓
Building features for a classifier or clustering articles by topic	✓
Summarising what a time period was "about" relative to a background corpus	Partially — see below

The partial exception: keyness as pseudo-TF-IDF

Some researchers compute a **keyness score** by treating the recent period as the "document" and the reference period as the "corpus" — which is conceptually similar to TF-IDF. But the statistically rigorous version of this is exactly the log-likelihood + log ratio approach already described. G^2 is essentially the principled, significance-tested version of what TF-IDF approximates intuitively.

Bottom line

For your goal — words that shifted significantly between week 5 and weeks 1-4 — G^2 with log ratio is the right tool. TF-IDF would give you a rough proxy at best and a misleading ranking at worst.

The `corpus_stats()` function contains the following comment:

```
#  $G^2 = 2 * \sum(\text{observed} * \log(\text{observed} / \text{expected}))$ 
```

Is this the formula to calculate the log-likelihood ratio? What do the observed and expected variables contain?

Yes, that's the correct formula for G^2 . Let me break it down.

The full formula

$$G^2 = 2 \times \sum O_i \times \ln(O_i / E_i)$$

The sum runs over all **four cells** of the contingency table:

	Recent (week 5)	Reference (weeks 1-4)
Target word	a	b
All other words	c	d

Observed values

Simply the raw counts from your data:

$$\begin{aligned}
 O_1 &= a \quad (\text{word in recent}) \\
 O_2 &= b \quad (\text{word in reference}) \\
 O_3 &= c \quad (\text{all other words in recent}) \\
 O_4 &= d \quad (\text{all other words in reference})
 \end{aligned}$$

Expected values

The counts you would *expect* if the word's relative frequency were identical in both corpora — i.e. under the null hypothesis that nothing changed. They're computed from the row and column marginals:

$$\begin{aligned}
 E_1 &= (a + b) \times (a + c) / N \\
 E_2 &= (a + b) \times (b + d) / N \\
 E_3 &= (c + d) \times (a + c) / N \\
 E_4 &= (c + d) \times (b + d) / N
 \end{aligned}$$

Where $N = a + b + c + d$ is the total number of tokens across both corpora.

Concrete example

Say your recent corpus has 10,000 tokens and your reference has 40,000 tokens, and the word "crisis" appears 30 times recently and 60 times in the reference:

$$\begin{aligned}
 a &= 30, & b &= 60 \\
 c &= 9970, & d &= 39940 \\
 N &= 50000 \\
 \\
 E_1 &= (30+60) \times (30+9970) / 50000 = 90 \times 10000 / 50000 = 18 \\
 E_2 &= (30+60) \times (60+39940) / 50000 = 90 \times 40000 / 50000 = 72 \\
 E_3 &= (9970+39940) \times 10000 / 50000 = 49910 \times 10000 / 50000 = 9982 \\
 E_4 &= (9970+39940) \times 40000 / 50000 = 49910 \times 40000 / 50000 = 39928 \\
 \\
 G^2 &= 2 \times (30 \times \ln(30/18) + 60 \times \ln(60/72) + 9970 \times \ln(9970/9982) + \\
 &\quad 39940 \times \ln(39940/39928))
 \end{aligned}$$

$$\approx 2 \times (16.2 - 9.9 - 1.2 + 1.2)$$
$$\approx 12.7$$

"crisis" appears at 3× its expected relative frequency (30/18), and $G^2 = 12.7$ exceeds the chi-squared critical value of 3.84 at $p < 0.05$ with 1 degree of freedom, so the increase is significant.

