NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Ideas in Context: Measuring Party-Group Convergence and Divergence in EU Debates

Kyriakos Lampros G. Kiouranas

Supervisor: **Manolis Koubarakis, Professor**

ATHENS

September 2025

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ιδέες στο πλαίσιο: Μέτρηση της σύγκλισης και της απόκλισης κομμάτων-ομάδων στις συζητήσεις της ΕΕ

Κυριάκος Λάμπρος Γ. Κιουράνας

Επιβλέπων: **Μανόλης Κουμπαράκης, Καθηγητής**

ΑΘΗΝΑ

Σεπτέμβριος 2025

BSc THESIS

Ideas in Context: Measuring Party-Group Convergence and Divergence in EU Debates

Kyriakos Lampros G. Kiouranas
S.N.:1115201900238

SUPERVISOR: **Manolis Koubarakis, Professor**

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ιδέες στο πλαίσιο: Μέτρηση της σύγκλισης και της απόκλισης κομμάτων-ομάδων στις συζητήσεις της ΕΕ

Κυριάκος Λάμπρος Γ. Κιουράνας
Α.Μ.: 1115201900238

ΕΠΙΒΛΕΠΟΝΤΕΣ: Μανόλης Κουμπαράκης, Καθηγητής

ABSTRACT

I built an end-to-end pipeline to track how European Parliament party groups talk about one idea over time (1999–2023), for example, security, health, or trade. The steps are, take the official speech texts [1], keep only the parts that truly discuss the chosen idea, turn those sentences into numbers that capture meaning, and measure how close or far party groups are in what they say. If groups are near, they frame the idea similarly; if far, they use different frames. I work at the sentence level because one speech often mixes topics. For each theme, I scan sentence by sentence and keep only those matching carefully chosen keywords. This yields focused text for each party group and time point (by year or parliamentary term). I then create vector embeddings with a modern transformer model [2][3]. In plain terms, the model reads a sentence and produces a numeric “fingerprint.” When needed, I also embed a specific word (e.g., “security” or “COVID”) inside its sentence to capture context, since the same word can mean different things across policy areas [4]. I aggregate many sentence vectors into one representative vector per party group and time point. I also run KMeans to find meaningful clusters within each party/time slice. To measure differences between groups, I use cosine distance (0 = very similar, 1 = very different), treating it as a polarization score. I repeat the same pipeline across themes, economy, energy, environment, health, immigration, security/defence, trade, transportation, plus digital, enlargement, integration, and war, so results are comparable. For visuals, I produce (1) timelines where each line shows distance between party pairs over time (good for exact shifts and spikes), and (2) t-SNE maps that place parties on a 2-D plane (good for intuition, axes not interpretable) [6]. The pipeline is modular and reproducible. Each step runs from the command line, writes its own files, keeps detailed logs (selected sentences, included groups/years, thin data flags), and caches intermediate outputs for faster reruns. All outputs are saved per theme with clear names.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: European Parliament, Party Groups, Political Discourse, Semantic Polarization, Contextual Embeddings

ΠΕΡΙΛΗΨΗ

Έφτιαξα ένα end-to-end pipeline για να παρακολουθώ πώς οι πολιτικές ομάδες του Ευρωπαϊκού Κοινοβουλίου μιλούν για μια ιδέα στον χρόνο (1999–2023), π.χ. ασφάλεια, υγεία ή εμπόριο. Τα βήματα είναι, παίρνω τα επίσημα κείμενα ομιλιών [1], κρατώ μόνο τα μέρη που όντως συζητούν την επιλεγμένη ιδέα, μετατρέπω αυτές τις προτάσεις σε αριθμούς που αποτυπώνουν το νόημά τους και μετρώ πόσο κοντά ή μακριά είναι οι ομάδες μεταξύ τους σε αυτά που λένε. Αν οι ομάδες είναι κοντά, πλαισιώνουν την ιδέα με παρόμοιο τρόπο αν είναι μακριά, τη πλαισιώνουν διαφορετικά. Δουλεύω σε επίπεδο πρότασης, γιατί μία ομιλία συχνά αναμιγνύει θέματα. Για κάθε θεματική, «σκανάρω» πρόταση προς πρόταση και κρατώ μόνο όσες ταιριάζουν σε προσεκτικά επιλεγμένες λέξεις-κλειδιά. Έτσι έχω στοχευμένο κείμενο για κάθε πολιτική ομάδα και χρονική στιγμή (ανά έτος ή κοινοβουλευτική περίοδο). Στη συνέχεια δημιουργώ διανυσματικές ενσωματώσεις (embeddings) με ένα σύγχρονο μοντέλο transformer [2][3]. Με απλά λόγια, το μοντέλο «διαβάζει» μια πρόταση και παράγει ένα αριθμητικό «αποτύπωμα». Όταν χρειάζεται, ενσωματώνω και μια συγκεκριμένη λέξη (π.χ. «ασφάλεια» ή «COVID») μέσα στην πρότασή της για να πιάσω το συμφραζόμενο, αφού η ίδια λέξη μπορεί να χρησιμοποιείται αλλιώς σε διαφορετικές πολιτικές περιοχές [4]. Συγκεντρώνω πολλά διανύσματα προτάσεων σε ένα αντιπροσωπευτικό διάνυσμα ανά ομάδα και χρονική στιγμή. Τρέχω επίσης KMeans για να βρω νοηματικά clusters μέσα σε κάθε «φέτα» ομάδας/χρόνου. Για να μετρήσω διαφορές μεταξύ ομάδων, χρησιμοποιώ την απόσταση συνημιτόνου (cosine distance: 0 = πολύ παρόμοια, 1 = πολύ διαφορετική), την οποία αντιμετωπίζω ως δείκτη πόλωσης. Επαναλαμβάνω το ίδιο pipeline σε θεματικές, όπως οικονομία, ενέργεια, περιβάλλον, υγεία, μετανάστευση, ασφάλεια/άμυνα, εμπόριο, μεταφορές, καθώς και digital, διεύρυνση, ενοποίηση και πόλεμος, ώστε τα αποτελέσματα να είναι συγκρίσιμα. Για τα οπτικά, παράγω (1) χρονογραμμές όπου κάθε γραμμή δείχνει την απόσταση ανάμεσα σε ζεύγη ομάδων στον χρόνο (χρήσιμες για ακριβείς μεταβολές και αιχμές) και (2) t-SNE χάρτες που τοποθετούν τις ομάδες σε δισδιάστατο επίπεδο (καλοί για διαίσθηση, οι άξονες δεν ερμηνεύονται άμεσα) [6]. Το pipeline είναι δομοστοιχειωτό και αναπαραγώγιμο. Κάθε βήμα τρέχει από τη γραμμή εντολών, γράφει τα δικά του αρχεία, κρατά αναλυτικά logs (ποιες προτάσεις επιλέχθηκαν, ποιες ομάδες/χρόνια περιλήφθηκαν, πού τα δεδομένα ήταν φτωχά) και κάνει cache ενδιάμεσα αποτελέσματα για πιο γρήγορες επαναλήψεις. Όλες οι έξοδοι αποθηκεύονται ανά θεματική με σαφή ονόματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ευρωπαϊκό Κοινοβούλιο, Πολιτικές Ομάδες, Πολιτικός Λόγος, Σημασιολογική Πόλωση, Ενσωματώσεις Κειμένου

CONTENTS

LIST OF FIGURES

Preface

This thesis grew out of a simple question, which was how political groups in the European Parliament talk about the same idea across different policy areas and across time and how we can measure those shifts in a transparent, reproducible way. To answer it, I built an end-to-end pipeline that works at the sentence level, represents meaning with modern language models, and summarizes group positions with clear metrics and visuals. The aim is practical, it is to give readers tools to see convergence and divergence in context rather than rely only on counts or votes.

The project stands alongside classic approaches such as roll-call analysis and topic modeling. Roll-calls show how groups vote and topic models reveal broad themes. Here, I focus on framing how similar words carry different meanings in energy, immigration, environment, or security debates and I quantify those differences with semantic distance. The result is a set of comparable timelines and maps that make change over time visible and auditable.

Reproducibility guided every decision. Each step of the pipeline writes its own outputs, logs key choices, and caches intermediate files so analyses can be paused, resumed, and checked. Figures use consistent naming and the structure is modular, like models can be swapped, themes added and run repeatedly by year or by term. The data are official parliamentary speeches and the methods respect the public nature of the material while documenting filters and thresholds to avoid thin or misleading cells.

1. INTRODUCTION

In this thesis, I study how the party groups in the European Parliament talk about the same idea in different policy areas and over time. By “idea,” I mean a focal concept such as **security**, **health** or **trade**. I aim to move beyond counting words and instead look at meaning in context. [5] To do that, I turn sentences from parliamentary speeches into numbers that capture their meaning, compare party groups using those numbers and track how close or far their language is from each other across years and themes. I also make the results easy to see with clear charts and simple maps.

1.1 Objective

My objective is to build a clear, repeatable pipeline that shows how party groups frame the same idea and how that framing changes over time. I want to:

- collect the parts of speeches that truly talk about the idea I’m studying,
- turn those sentences into meaningful numeric representations,
- summarize each party group’s position at each point in time and
- measure how similar or different the groups are from each other.

In short, I want a reliable way to see who is close, who is far and when those distances change.

1.2 Motivation

Why does this matter? In politics, the words parties use can signal priorities, values, and policy choices. Two groups might both say “security,” but one may mean border control while another may mean energy supply or digital safety. If I only count the words, I miss the nuance. If I look at the word in its sentence, I can capture what it actually means in that moment.

This is also useful for policy evaluation. If groups move closer in how they talk about an issue, that can suggest growing agreement, if they move apart, it can signal tension or competing agendas. Tracking these shifts over time helps put debates in context (for example, around major events like the pandemic or war) and gives a grounded view of how language reflects political change.

1.3 Overview

Here is how my approach works, in simple terms:

1. **Focused text selection (Preprocessing).**
   I start with official European Parliament speeches.[1] Instead of using whole speeches, I scan them sentence by sentence and keep only the sentences that match the topic I care about (using well-chosen keywords and their variants). This gives me a clean, focused set of sentences per party group and per time period (year or parliamentary term).

2. **Meaning as numbers (Embeddings).**
   I use a modern language model to turn each sentence into a vector list of numbers that captures its meaning. [2] [3] If I need to, I can also zoom in on a specific word (like “security”) inside a sentence and build a vector just for that word in that exact context. This is important because the same word can mean different things in different policy areas.

3. **Group summaries (Aggregation).**
   I combine many sentence vectors to get one representative vector for each party group at each point in time. You can think of this as the group’s “average position” on the idea during that period.

4. **Measuring distances (Metrics & visuals).**
   I compare party groups by measuring the distance between their vectors. I use a standard measure (cosine distance) where a small number means “the groups talk about the idea in similar ways” and a large number means “they talk about it differently.”
   I then visualize the results in two ways:

   - Timelines that show how distances change over time, making turning points and spikes easy to spot.
   - Simple 2-D maps (using t-SNE) that place groups as points on a plane so that nearby points mean similar language and farther points mean different language. These maps are for visual intuition, while the timelines give the exact numbers.

The whole pipeline is modular and reproducible. I can run each step from the command line, every step saves its results and all actions are logged. That means I can pause, resume and audit what happened at any time.

1.4 Thesis Structure

The thesis follows a straightforward path:

- **Introduction (Chapter 1).**

  I set the scene, like what I want to learn, why it matters, and how I plan to study party-group language over time and across policy areas.

- **Background (Chapter 2).**

  I explain, in plain words, how I turn text into numbers, why context matters for meaning, and how I measure differences between party groups.

- **Data and Pipeline Architecture (Chapter 3).**

  I describe my data source, how I pick the right sentences, how I build embeddings, how I summarize groups and how I save logs and results, so everything is reproducible.

- **Analysis Procedure (Chapter 4).**

  I show exactly how I compute relative and absolute polarization, how I make t-SNE maps, and how I build word clouds to keep results human-readable.

- **Results (By Context) (Chapter 5).**

  I apply the pipeline to each theme (baseline, economy, energy, environment, health, immigration, security, defence, trade, transportation, digital, enlargement, integration, war). I compare patterns across time and end with a cross-context takeaway.

- **Robustness, Caveats, and Interpretability (Chapter 6).**

  I explain how to read the maps safely, how I handle small data cells and when I use clustering versus simple averages, so conclusions stay reliable.

- **Reproducibility Guide (Chapter 7).**

  I give copy-paste commands for single-theme runs, all-themes runs, and grouped-keyword runs, plus tips on options and where outputs are saved.

- **Related Methods (Chapter 8).**

  I place my approach next to common alternatives (bag-of-words, dictionaries, topic models, static embeddings) and explain why I use contextual transformers.

- **File Inventory (Chapter 9).**

  I list every key script and helper, what its inputs/outputs are and how the pieces fit together from Phase 1 to final figures.

- **How to Read the Figures (Quick Guide) (Chapter 10).**

  I give a short guide for interpreting polarization timelines, t-SNE maps and comparative panels.

- **Conclusion (Chapter 11).**

  I summarize what I found, like a shared core meaning with theme-specific splits (notably immigration mid-period and environment and economy more recently) and a narrowing gap in hard-security language.

- **Bibliography.**

  I list all sources and datasets used.

This structure keeps the writing simple and the workflow transparent, so the results are both understandable and repeatable.

2. BACKGROUND

2.1 Political text analysis and semantic representation

Most traditional studies of political speech use bag-of-words or dictionary methods. [7] In a bag-of-words, I count how often each word appears and ignore word order. In dictionary methods, I look for words from a hand-built list (for example, words tied to “security” or “economy”) and count those. These tools are useful, but they have clear limits. They don’t know that “border security” and “energy security” use the same word in very different ways. They also struggle with synonyms (e.g., migrant vs asylum seeker) and with languages where the same idea can appear in many forms.

To go beyond raw counts, I need a way to represent meaning. That is where embeddings come in. An embedding is just a vector, a short list of numbers, that represents a piece of text. Two pieces of text that “mean” similar things should have similar vectors (they sit close together in this numeric space). Older embeddings (like classic word vectors) give one vector per word, no matter what the context is. [7] Newer models (like RoBERTa or MiniLM) are contextual, which means that they read the whole sentence and adjust the representation of each word based on its neighbors. [2] [3] This is crucial for political language, where the same term can be used strategically or with different policy lenses.

In my work, I use these contextual models so that my numbers reflect how a party group is using a word in a specific sentence, not just the word by itself. This lets me compare parties more honestly, I’m comparing meaning in context, not just word counts.

2.2 Contextualized embeddings and token-level extraction

When I say “contextual,” I mean the model looks at the entire sentence (and sometimes more) before producing a vector. I use two related views:

- Sentence embeddings. I turn a whole sentence into one vector. [4] This works well when I want a broad picture of how a group talks about a theme without focusing on any single word.
- Keyword-in-context embeddings. Sometimes I care about a specific word, like security, inside a sentence. In that case, I find the exact place where that word appears in the model’s tokenized input (the model splits text into sub-tokens). [2] [3] I then pool (average) the hidden states for those tokens to get a vector for that word in that sentence. If the match is messy (because the word got split into pieces), I use safe fallbacks, so I still get a stable representation.

Why do I bother with word-in-context? Because “security of supply” (energy policy) and “border security” (immigration policy) should not look the same. Token-level extraction

preserves that nuance. If I do not specify a keyword, I fall back to sentence-level embeddings, which are great for a quick, theme-level overview.

This two-track setup gives me flexibility. I can study broad attitudes with sentence vectors and then zoom in on the exact word to see how its meaning shifts across policy areas and over time.

2.3 Polarization metrics and semantic geometry

Once I have vectors for each party group and time period, I need a clear way to say how close or far groups are from one another in how they talk. I use cosine distance for this. Picture two arrows from the origin of a graph, like, for example, the cosine distance tells me how different their directions are. If the distance is small, the groups speak in similar ways. If it is large, they speak differently.

I look at polarization in two complementary ways:

Relative polarization (group-to-group).
I compute distances between every pair of party groups for a given theme and time. This tells me, for example, whether EPP is closer to S&D than to Greens/EFA on “security” each year. I then track these distances over time to see when pairs converge or diverge and whether big events line up with sudden changes.

Absolute polarization (distance-to-centroid).
I also compute the centroid, the simple average of all party vectors for that theme and time. Then I measure each group’s distance to this center. This shows who speaks like the mainstream and who is more of an outlier in each period.
Watching these distances over time highlights shifts in who leads or departs from the common conversation.

To make the overall picture easier to see, I project the high-dimensional vectors down to two dimensions using a technique called t-SNE. [6] On these maps, points that are near each other represent groups that use similar language; points that are far apart use different language. The maps help me spot clusters, outliers and movement from one period to the next.

One important note, though, the t-SNE maps are for intuition, not precise measurement. The timelines of distances are the hard numbers. I use the maps to guide the eye and the timelines to support the exact claims. Taken together, these tools give me a balanced view of what changed (distances) and how it looks (semantic neighborhoods) in the language of European Parliament party groups.

3. DATA AND PIPELINE ARCHITECTURE

3.1 Data sources and schema

- I work with a prepared dataset of European Parliament speeches covering 1999–2023. [1] The file is in RDS format (an R data file). I load it in Python with pyreadr, which lets me read the table without converting it by hand. Each row is a speech (or speech segment) with the fields I need for analysis. The most important fields are:
- the political group of the speaker (a short code I call epg_short),
- the time information (either a year or a parliamentary term, stored as term_no),
- the language/text of the speech,
- simple counts such as word count, which I can compute on the fly to filter very short items.
- I do not hard-code which political groups to analyze. Instead, I discover the groups from the data each time I run the pipeline. This way, if a group merges, splits or appears only in certain years, the analysis adjusts automatically. When I load the data, I log the size of the corpus, which years or terms are present and which groups I find, so I can check that everything looks right before I go further.
- Artifacts, like segmented sentences, per-theme caches, embeddings, analysis CSV/PNG reports.

3.2 Pre-processing and sentence segmentation (Phase 1)

My first step is to pull out only the parts of the text that really talk about the theme I'm studying (for example, security, health or trade). Whole speeches often mix many topics, so I work at the sentence level.

I split each speech into sentences and when useful, attach simple lemma or stem forms to words. This helps me catch different versions of the same idea (like secure, security, securing). I keep a curated list of keywords for every theme. These lists include obvious words (like asylum for immigration) and also domain terms (like Frontex, GDPR, COVID, market, tariffs and crime), so I get better recall. For robustness, I expand each keyword to its variants using stems/lemmas, then search sentences with both the base forms and the variants.

For every theme, I filter the corpus sentence by sentence and create a tidy slice of text for each (time, group) pair. To avoid noisy results, I set a minimum number of sentences per cell. If a party group barely mentions a theme each year or term, I drop that cell and record this decision in the logs. I cache the segmented and filtered sentences to disk (with a version tag), so I don't have to redo this step if I run the pipeline again with the same settings. These cached files live under the run folder and follow a clear name pattern like:

poc_<keyword-or-group>_<theme>_sentences.pkl.

3.3 Embedding extraction (Phase 2)

Once I have clean, theme-focused sentences, I turn them into vectors (lists of numbers) that capture their meaning. I use a transformer model for this. [2] [3] In everyday terms, the model reads a sentence and produces a compact numeric "fingerprint" that represents the sentence's meaning in its context.

When I study a specific word inside a sentence (for example, the word security in a sentence about energy or migration), I use a helper that finds the exact token positions of that word in the model's input and averages the hidden states for those tokens. If the model's subword tokenization makes an exact match tricky, I use safe fallbacks (for example, averaging all non-special tokens) rather than failing. This lets me compare the same word across different policy contexts in a stable way.

I keep the model setup pluggable. I can switch between, for example, RoBERTa and DistilRoBERTa to trade speed for accuracy. The command-line runner exposes a --model option, and later phases reuse the same interface. I batch sentences to use the GPU efficiently and split long inputs into manageable chunks to avoid memory issues. The output of this phase is a set of vector representations ready for aggregation.

3.4 Analysis & reporting (Phase 3)

With vectors in hand, I build group-level summaries and compute the metrics I need. I do this in two complementary ways:

Average pooling. I take all sentence vectors for a given party group and time and average them. This gives me a single, stable vector that represents how that group talked about the theme during that period. It's fast, simple, and works well as a baseline.

Clustering (optional). If I expect distinct sub-topics inside the same theme (for example, a group using "security" in both border and cyber senses), I run KMeans and let the number of clusters be dynamic. I try several values of K and pick the smallest one that passes a silhouette quality threshold. [8] I can then use the cluster centroids as richer summaries. When needed, I can also collapse clusters back to a single weighted vector so that all plots stay comparable.

Using these group-time vectors, I compute pairwise cosine distances between party groups. I treat these distances as a measure of polarization in language, like smaller numbers mean the groups speak about the idea in similar ways and larger numbers mean they frame it differently. I report the results in two main forms:

- Timelines of distances, so I can see when groups move closer or farther apart and how that changes across years or terms.
- Simple 2-D maps using t-SNE. [6] These maps place groups as points so that nearby points suggest similar language and far-away points suggest different language. I use these maps for visual intuition, and I rely on the timelines for exact comparisons (because t-SNE axes don't carry direct meaning).

For cross-theme comparisons, I sometimes build short theme concept vectors (for example, by embedding small templates like “A discussion about {keyword}.”) and averaging them. I cache these too, so I can reuse them across figures. The plotting layer keeps colors and styles consistent across themes and groups, and it saves figures with clear names, such as:

poc_polarization_over_time_<theme>.png

poc_semantic_space_tsne_<theme>.png

Each theme gets its own set of outputs, which makes it easy to compare and include in the thesis.

3.5 Runs, logging, and reproducibility

I control the whole process with a top-level runner that I can call from the command line. I choose the time unit (year or term), the themes I want, the model and other options. Every execution creates a timestamped run folder (for example, run_2025_09_14_1030/) that holds everything for that run:

- data/ for the cached, filtered sentences,
- embeddings/ for the vectors,
- analysis/ or visualizations/ for the timelines and maps,
- logs/ for detailed records of what happened,
- a small metadata file with the exact settings I used.

I log key facts at every step, like how many sentences I kept per time/group, which groups and years were included or dropped, what thresholds I applied and any warnings or errors. Because I cache intermediate results with version tags, I can resume a run where I left off or repeat the same run later and get the same outputs. This makes the pipeline auditable, repeatable, and friendly to experimentation and of course, I can also change one option (say, switch the model or the theme list), rerun and know exactly what changed and why.

4. ANALYSIS PROCEDURE

4.1 Relative polarization (group-to-group distances)

Here, I measure **how different** two-party groups are when they talk about the same theme in the same period (a year or a parliamentary term).

1. **Make one vector per group and time.**
   From earlier steps, I already have sentence vectors for each theme. I **average** those vectors to get **one representative vector** for each party group in each time period. You can think of this as the group's "position" on that theme during that time.
2. **Measure the gap between the two groups.**
   I compare two groups using **cosine distance**. In plain words, this tells me how much their "positions" point in the same direction.
   1. A **small** number means the groups talk about the theme in **similar** ways.
   2. A **large** number means they talk about it **differently**.
3. **Track change over time.**
   I compute these distances for all group pairs and **plot them on a timeline**. This lets me see when groups move closer (more agreement in language) or farther apart (sharper differences) and whether big events line up with spikes or dips.

4.2 Absolute polarization (distance-to-centroid)

Relative distances tell me how groups differ from each other. Here, I also ask, how far is each group from the overall center of the conversation in each period?

1. **Find the "center."**
   I take all the group vectors for a theme and time period and compute their centroid, the simple average of them all. This is the "semantic center" for that theme and time.

2. **Measure each group's distance to the center.**
   For every group, I compute the cosine distance to the centroid.

   - A small distance means the group is speaking in a way that's close to the overall conversation.

   - A large distance means the group is using language that's more distinct from everyone else.

3. **Plot the pattern.**
   I draw a chart of these "distance-to-center" values over time for each group. This shows who is consistently mainstream and who is often an outlier and when those roles change.

4.3 Dimensionality reduction (t-SNE semantic maps)

Numbers alone are hard to grasp, so I also provide a picture of the semantic space.

5. **From many numbers to two.**
   Each group-time vector can have hundreds of dimensions. I use t-SNE to map those vectors into two dimensions so I can plot them on a page. [6]

6. **How to read the map.**
   On the map, nearby points suggest similar language; far-apart points suggest different language. I label points by party group and time so I can see clusters and trajectories (for example, if a group drifts toward or away from others across periods).

4.4 Keyword-context word clouds and lexical signatures

To keep the results human and concrete, I also look at the actual words.

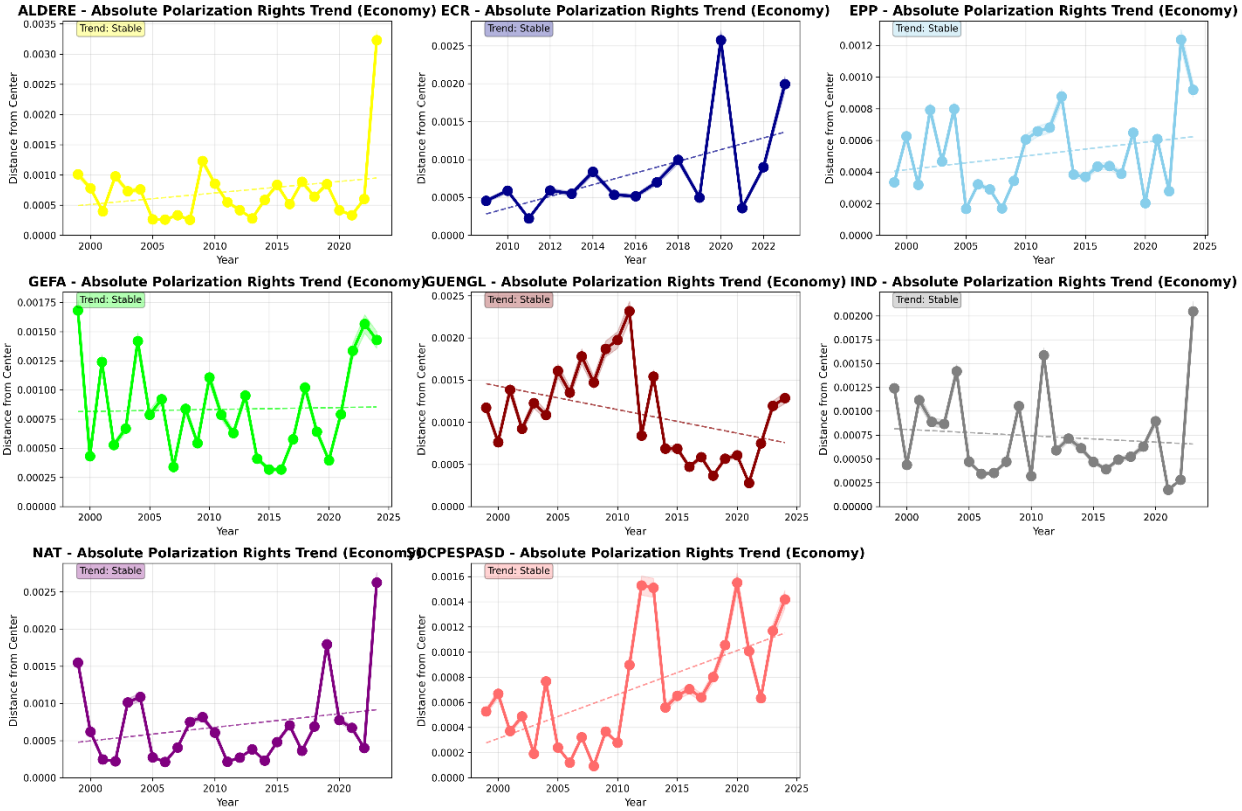
1. Collect the right text.
   For each theme, group and time period, I gather the filtered sentences that passed my keyword checks.

2. Clean and count.
   I remove stop words (very common words like “the,” “and,” “of”) and the focal keyword itself (so it doesn’t dominate). Then I count how often the remaining words appear.

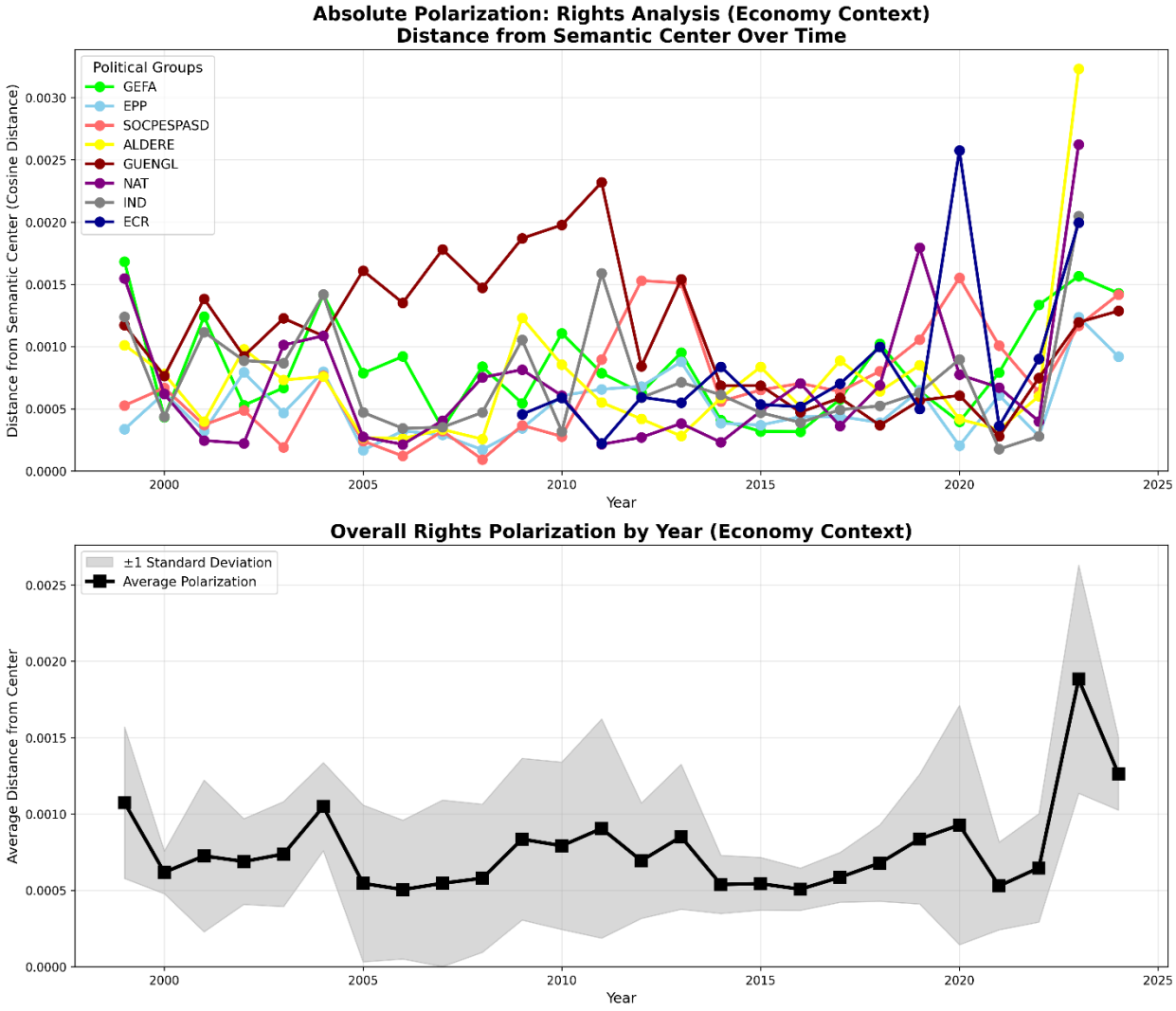
3. Build the word cloud.
   I create a word cloud where larger words are more frequent. This gives a quick sense of what each group emphasizes when they talk about the theme. Side by side, these clouds act like lexical signatures and help explain why certain groups cluster together or stand apart in the semantic space.

1. 5. RESULTS

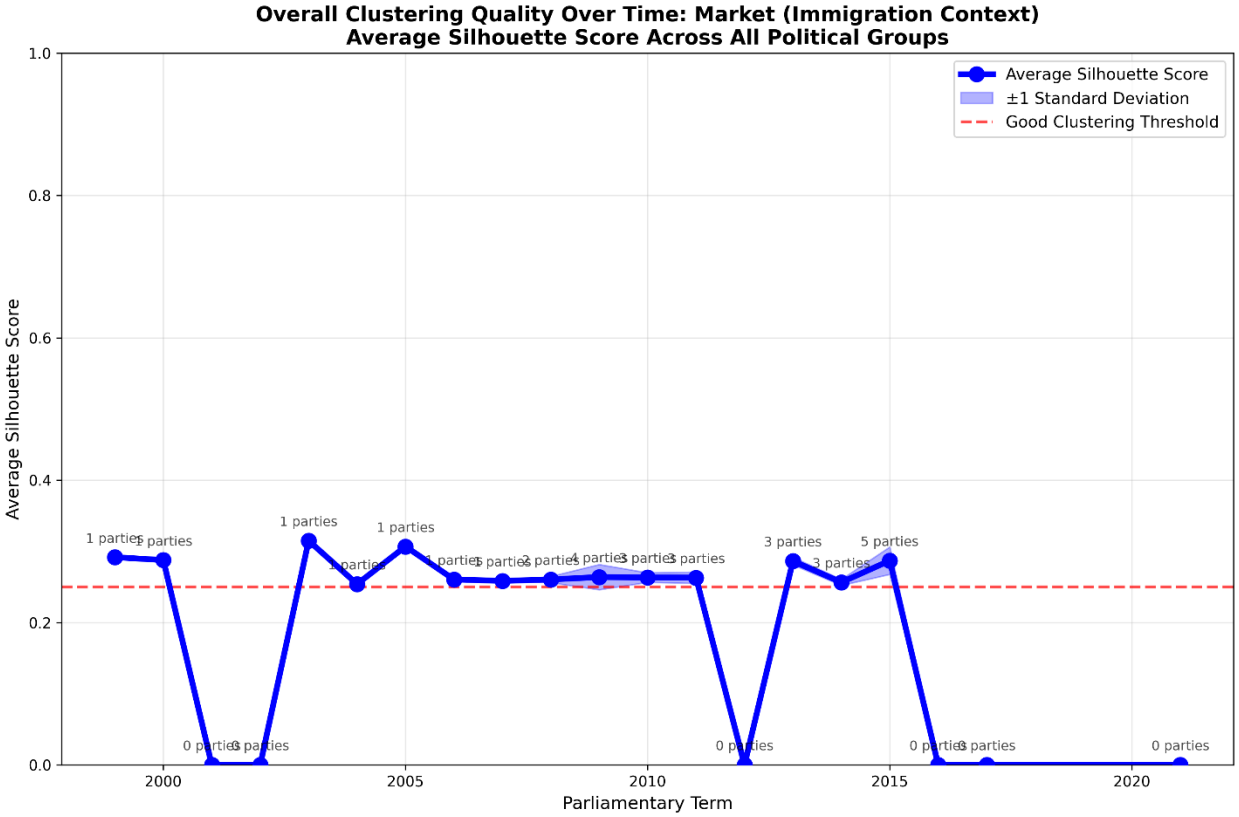
5.1 Economy



The top panel plots each group's yearly distance from the semantic center; the bottom panel shows the system average with ±1σ. Polarization is generally low-to-moderate, with a late-2010s/early-2020s uptick (notably an ECR spike ~2020 and ALDERE peak ~2023), before easing slightly.

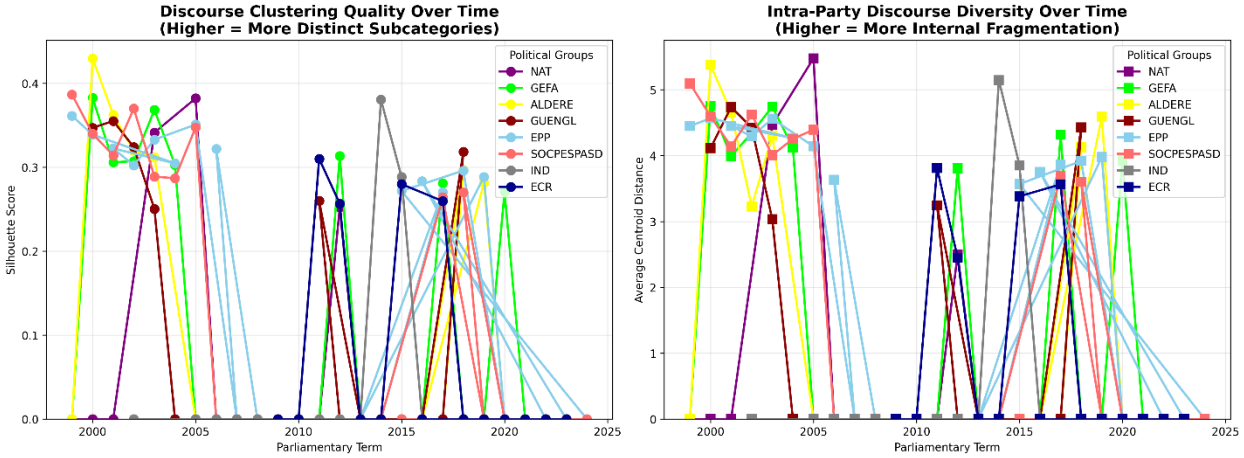


Lines are absolute distances with a light trend line per party. GUENGL trends gently downward overall, ECR exhibits post-2020 surges, ALDERE shows a sharp 2023 jump, and others move within a narrow band.

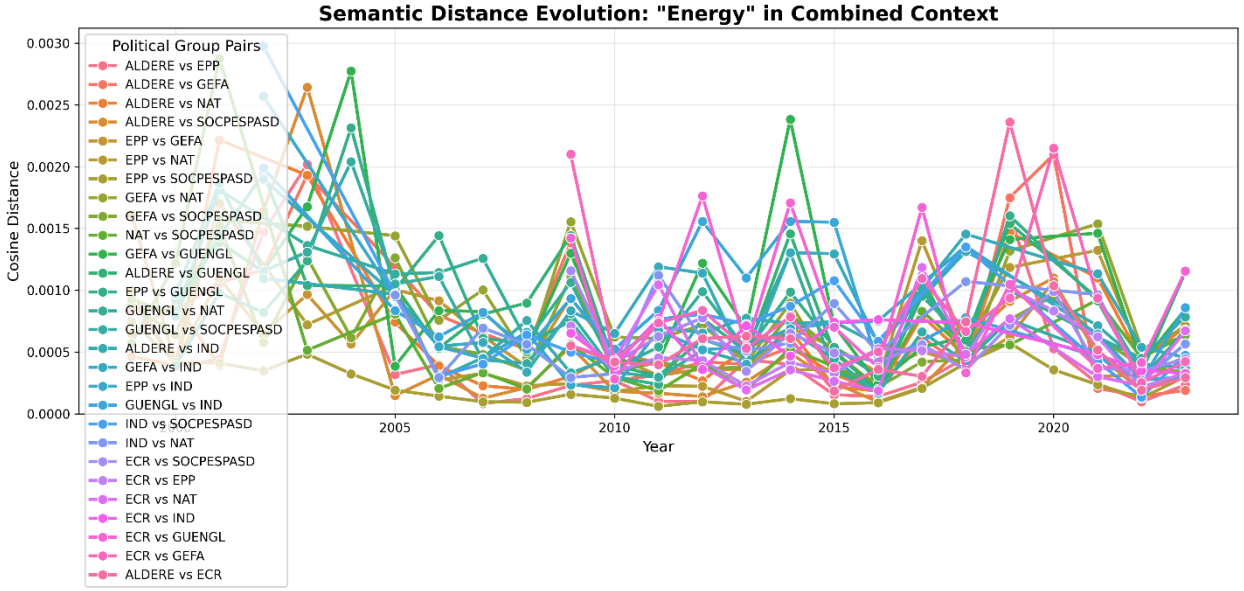


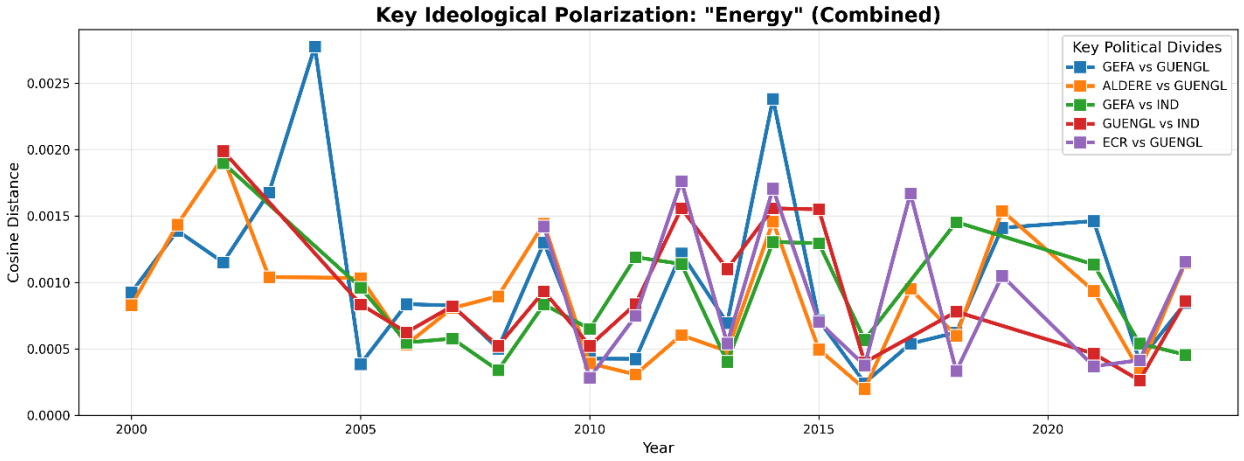
Market talk inside the immigration context is well-clustered for centrist groups, with moderate internal spread elsewhere.

5.2 Energy



Cross-party distances decline after 2005, with recurring mini-peaks (2013, 2019–2020). The most persistent divide: GEFA ↔ GUENGL



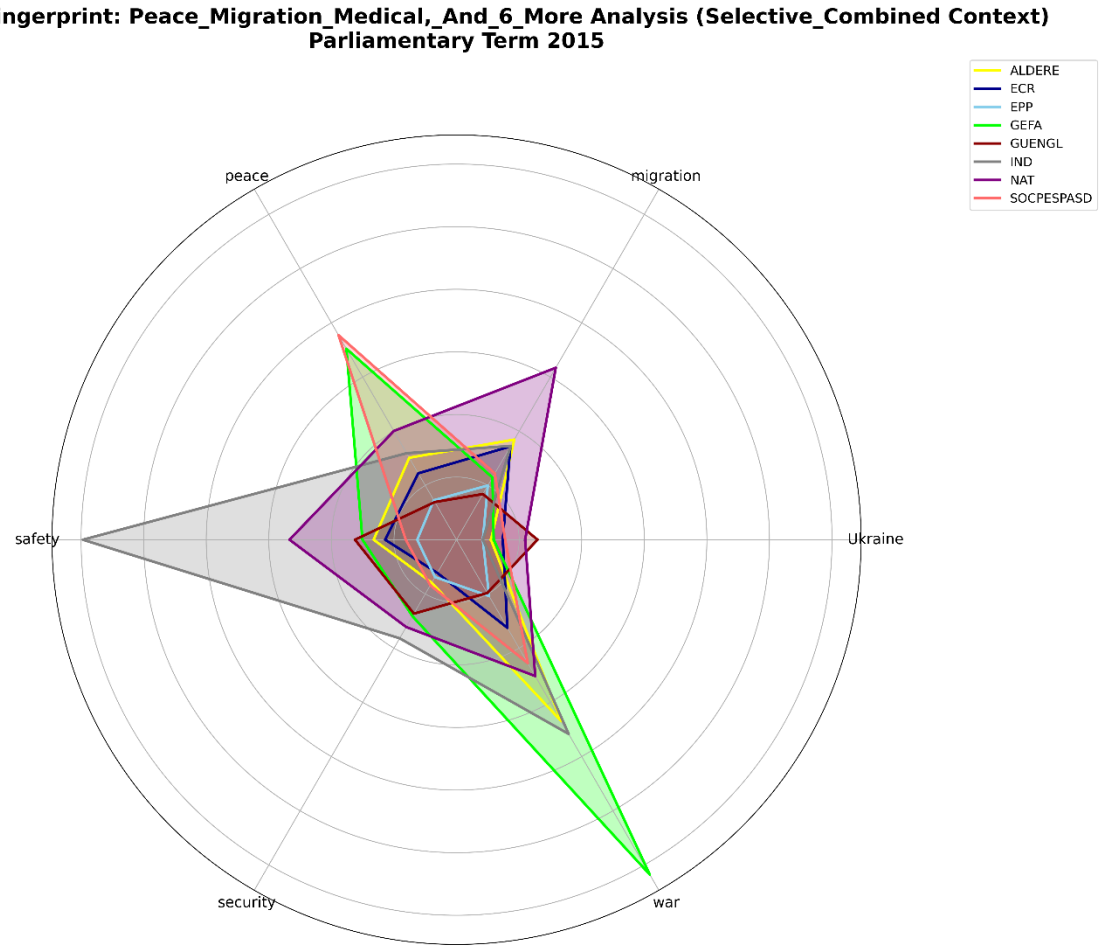


Polarization is cyclical; GEFA vs GUENGL/IND spikes recur, reflecting policy shocks

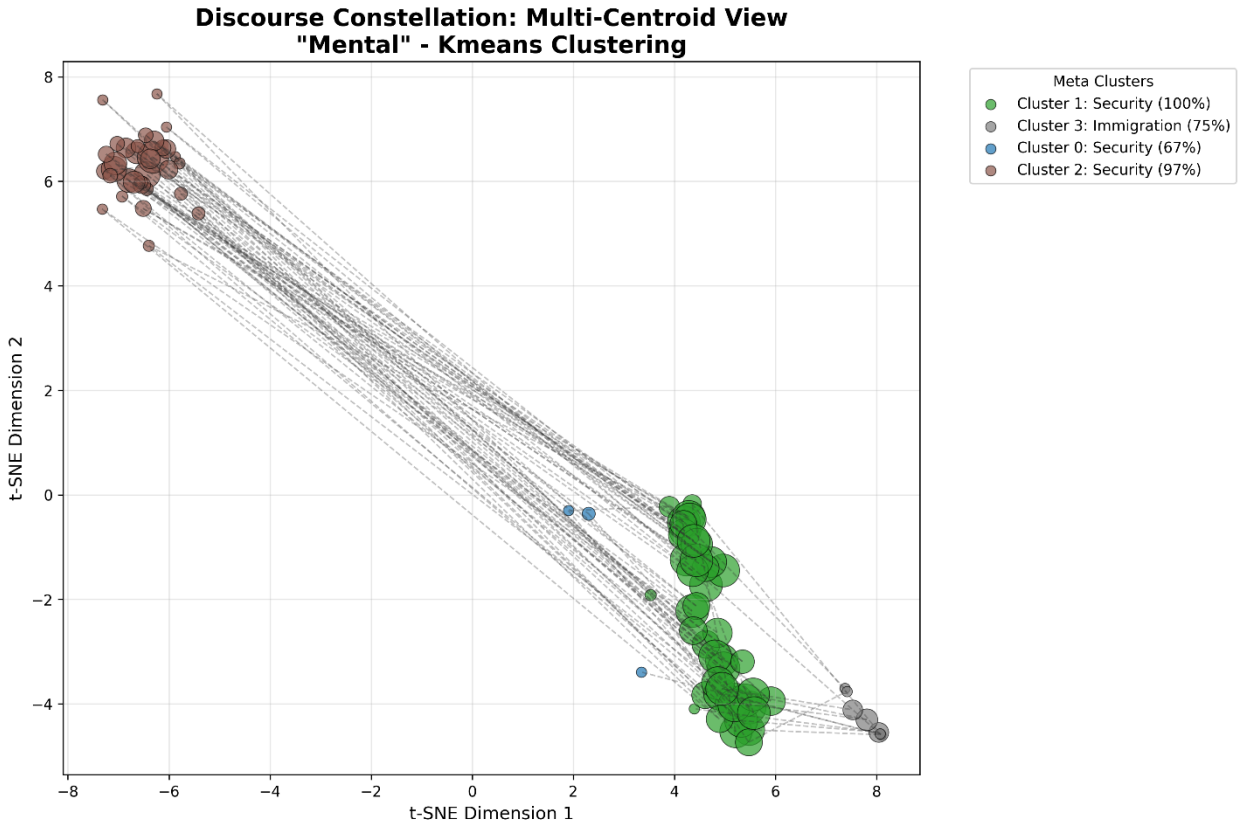
5.3 Environment

This map shows broad cross-party overlaps in how ‘security’ is framed when talking about the environment. Most party-years sit in a shared semantic core, with only a handful of outliers across the period (e.g., early-2000s and mid-2010s points drifting away from the center). The mixing of colors signals convergent vocabularies rather than hard ideological splits, so we should expect low-to-moderate polarization in the subsequent metrics and only episodic spikes.

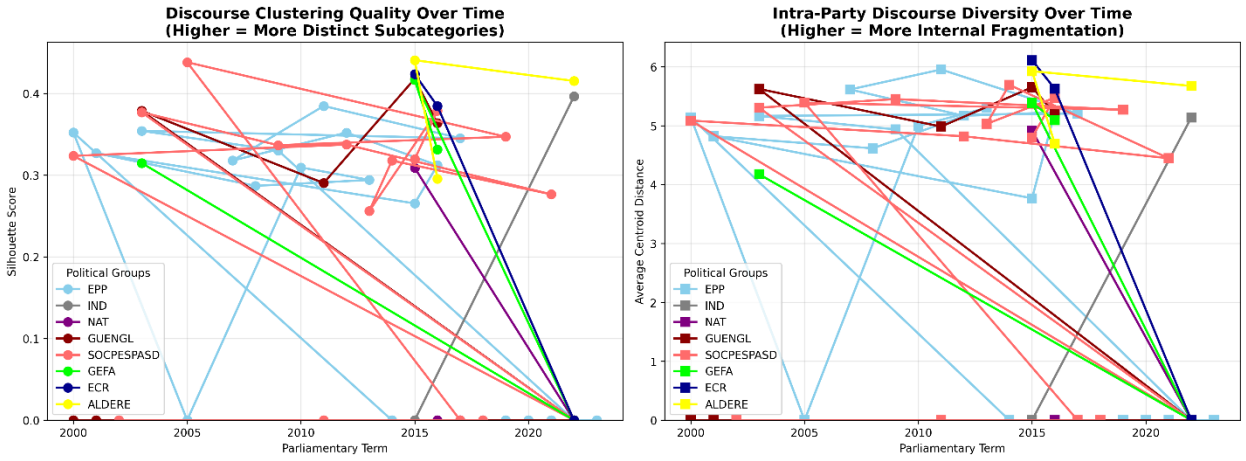
5.4 Health



Here, medical/health vocabulary is plotted alongside migration, safety, and war terms to show how health-security language is framed in Term 2015. The shape highlights whether groups emphasize public-health preparedness and safety or connect health to broader conflict/migration narratives.

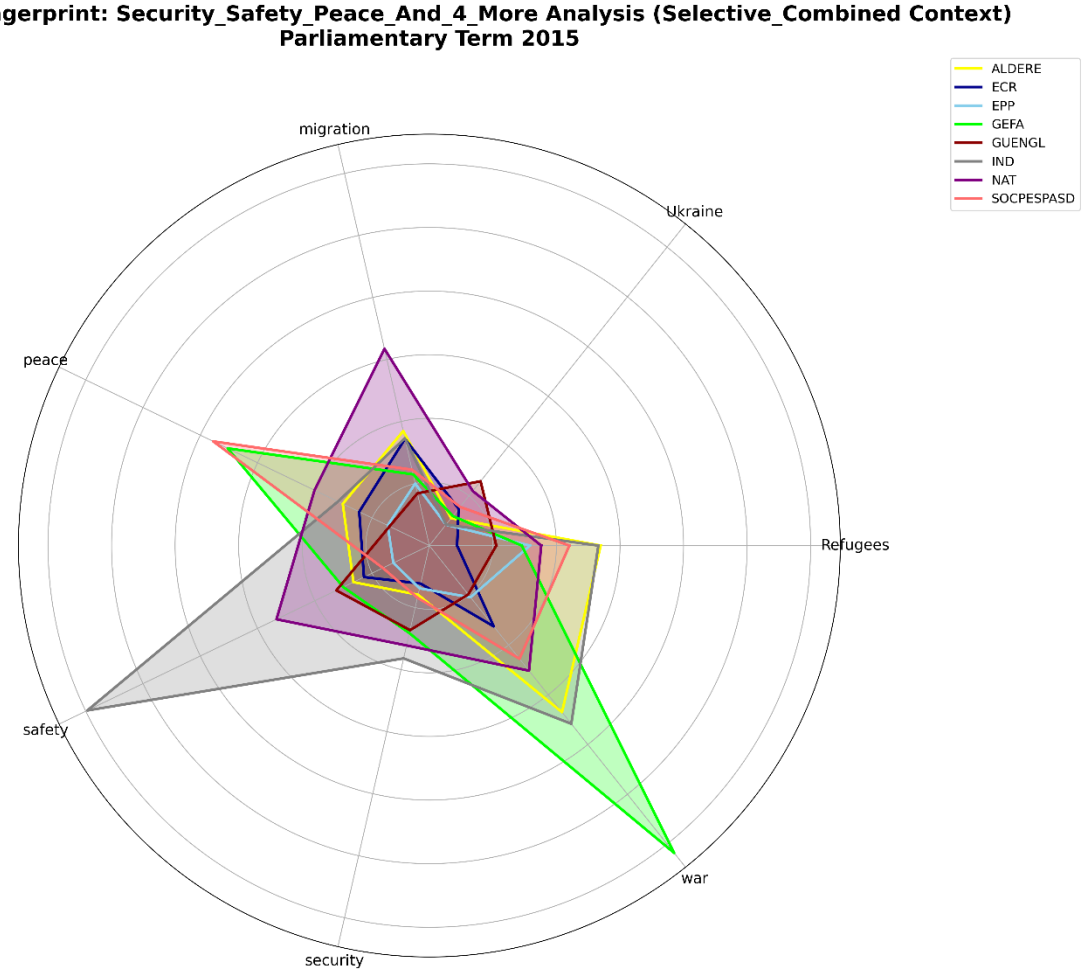


Allowing more centroids reveals a compact Security (100%) block plus a small Immigration (75%) pocket linking mental-health talk to integration/migration contexts. This shows how secondary frames can surface even in a health-first debate

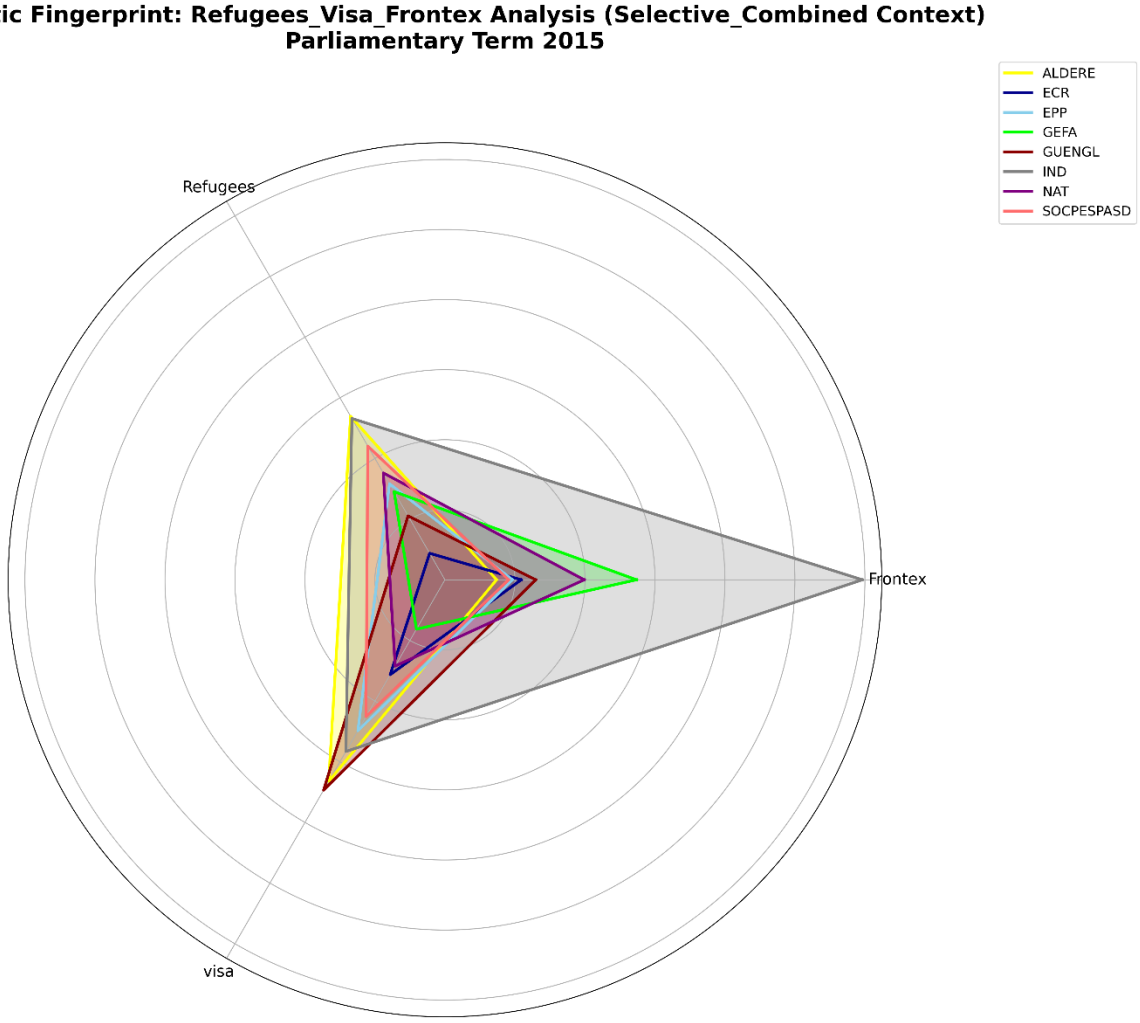


Most parties show modest sub-topic separation, punctuated by a few mid-2010s peaks (notably for ALDE/RE and ECR), implying episodic attempts to define sharper narratives (e.g., “war on disease,” health-security linkages). Internal diversity is uneven, some parties tighten their message in peak years, while others fragment.

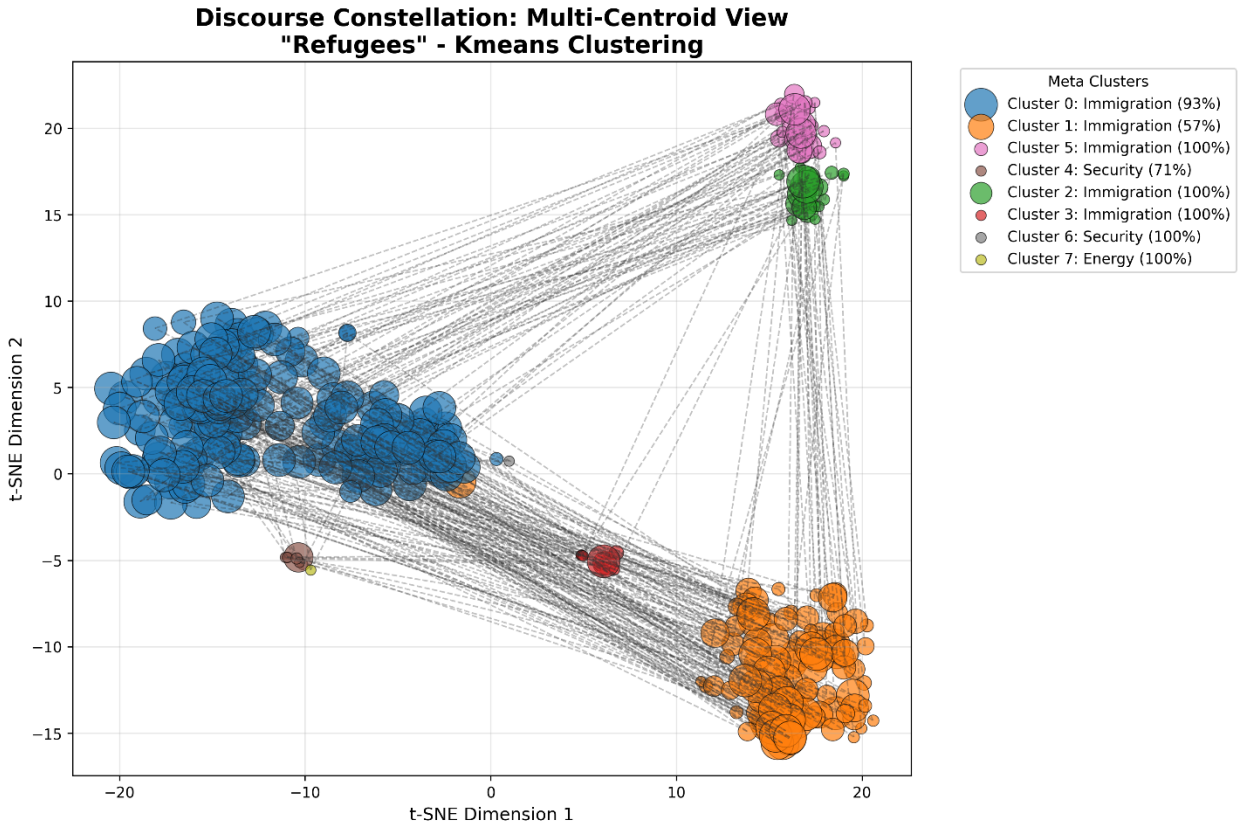
5.5 Immigration



This plot focuses on immigration discussions when they are framed through security, safety, and conflict. The “Refugees” and “Ukraine” spokes help signal humanitarian vs. geopolitical accents.

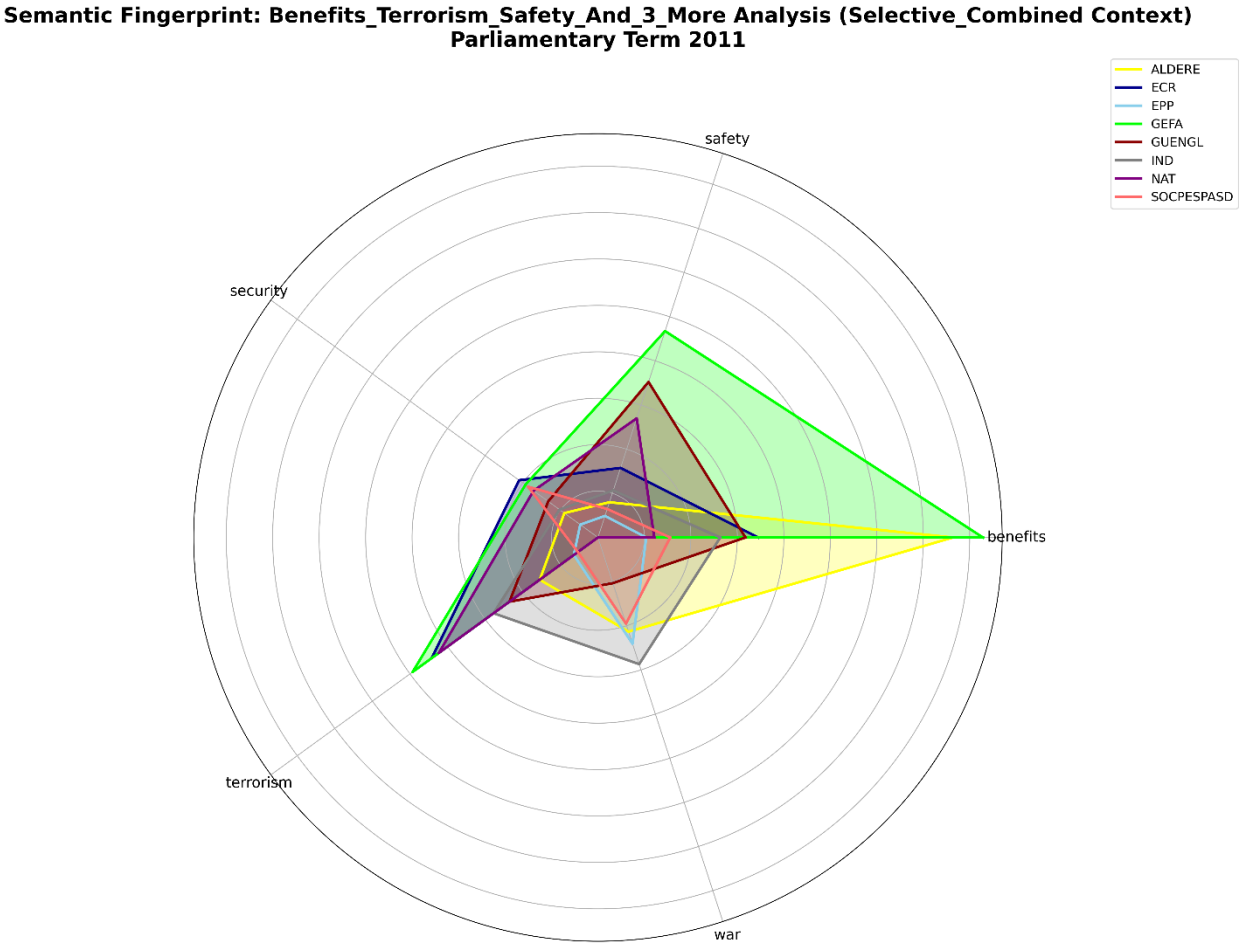


A focused view on border-management vocabulary. Higher values on “Frontex” and “visa” indicate stronger alignment with enforcement/administrative framing and higher “Refugees” points toward humanitarian/relocation emphasis.

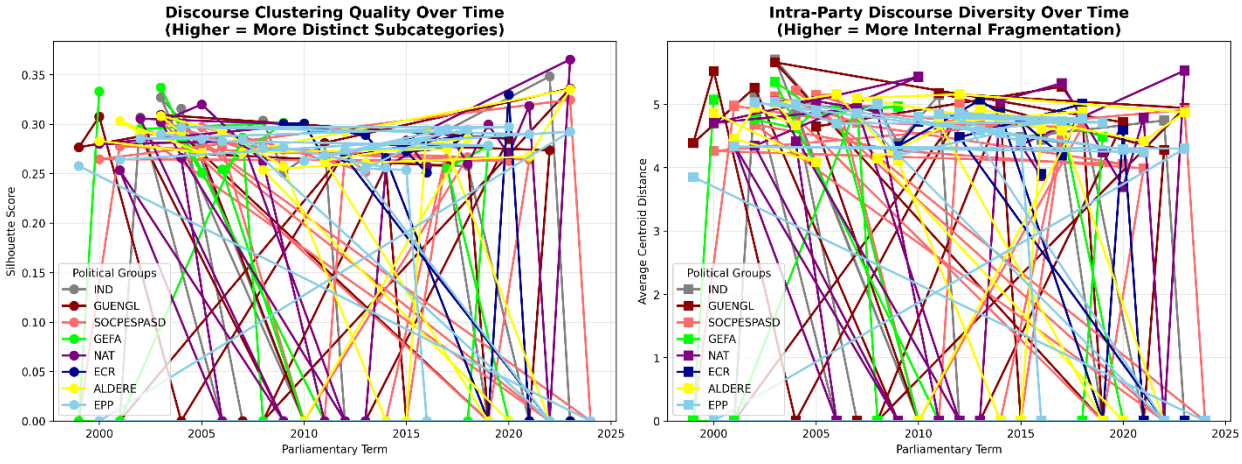


Refugees discourse splits into two big Immigration-coded neighborhoods (procedures, asylum, relocation) and a separate Security pocket (border/control), with a small Energy Island (supply/energy-shock tie-ins). This mix explains why immigration distances peak when humanitarian and enforcement frames pull apart.

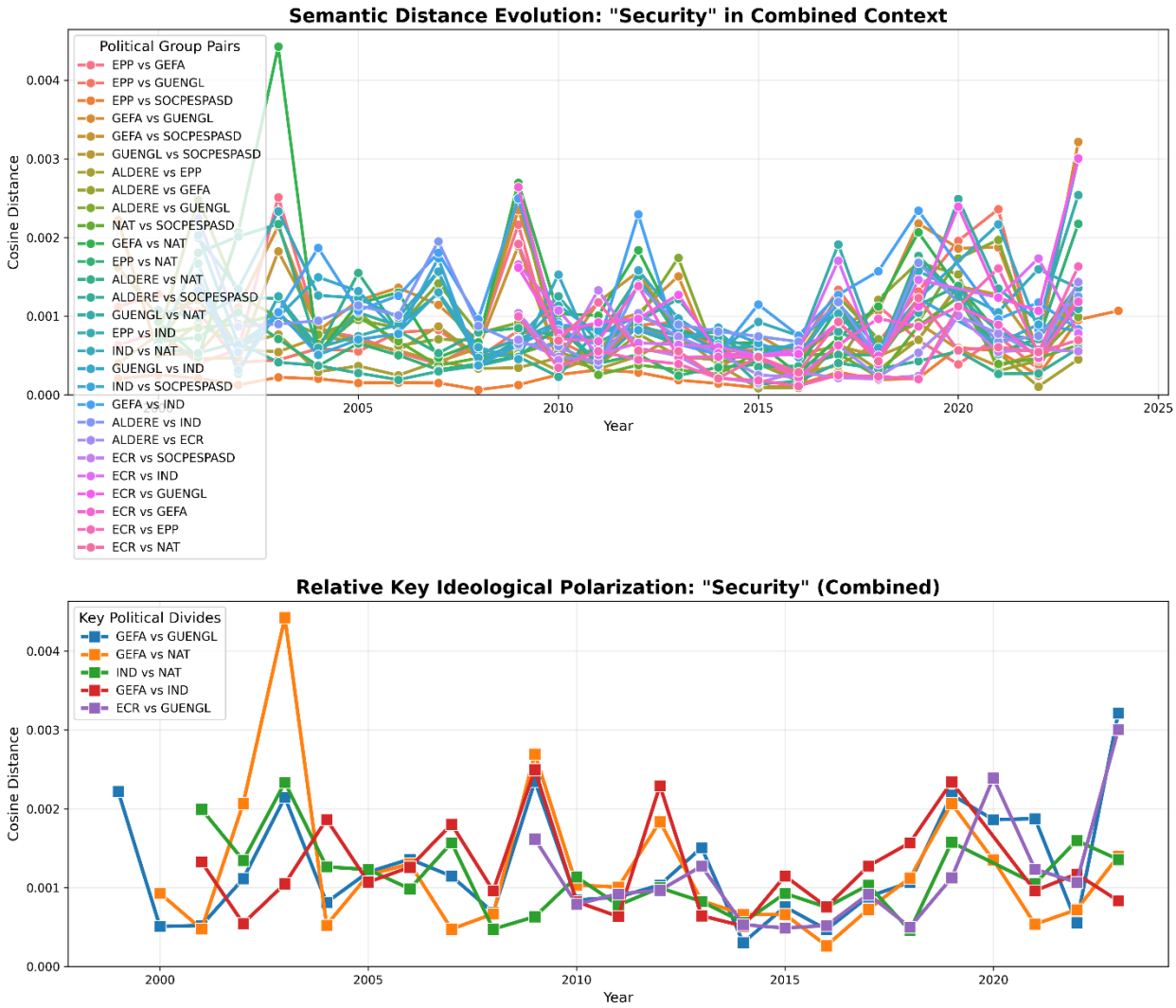
5.6 Security



This radar plot compares how party groups emphasize security-related notions in Term 2011. Each spoke is a keyword, a larger radio means higher normalized prominence within the security slice for that group. Together with the distance timelines, this offers a quick view of which groups leaned more toward safety/terrorism language versus broader “benefits” framing in that term.

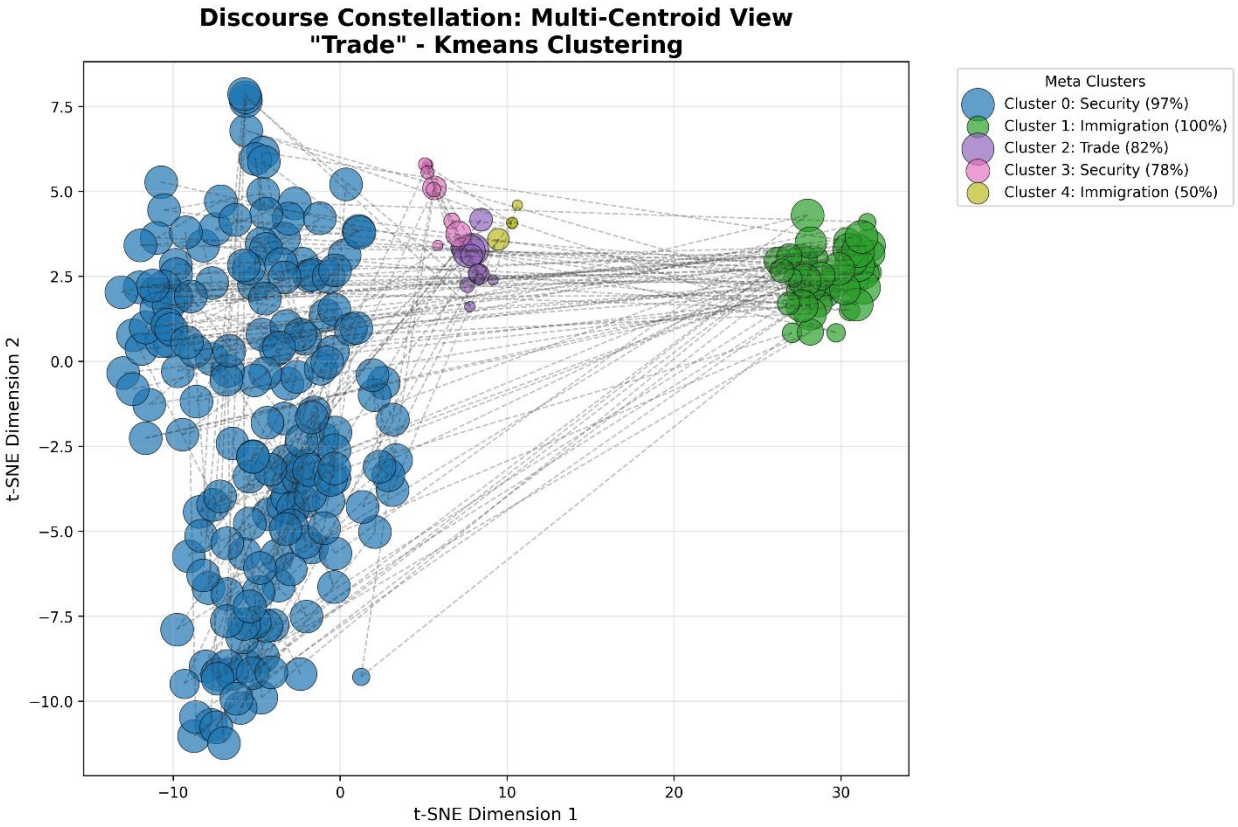


Security is one of the most structured spaces (left), but also shows sustained intra-party fragmentation (right).



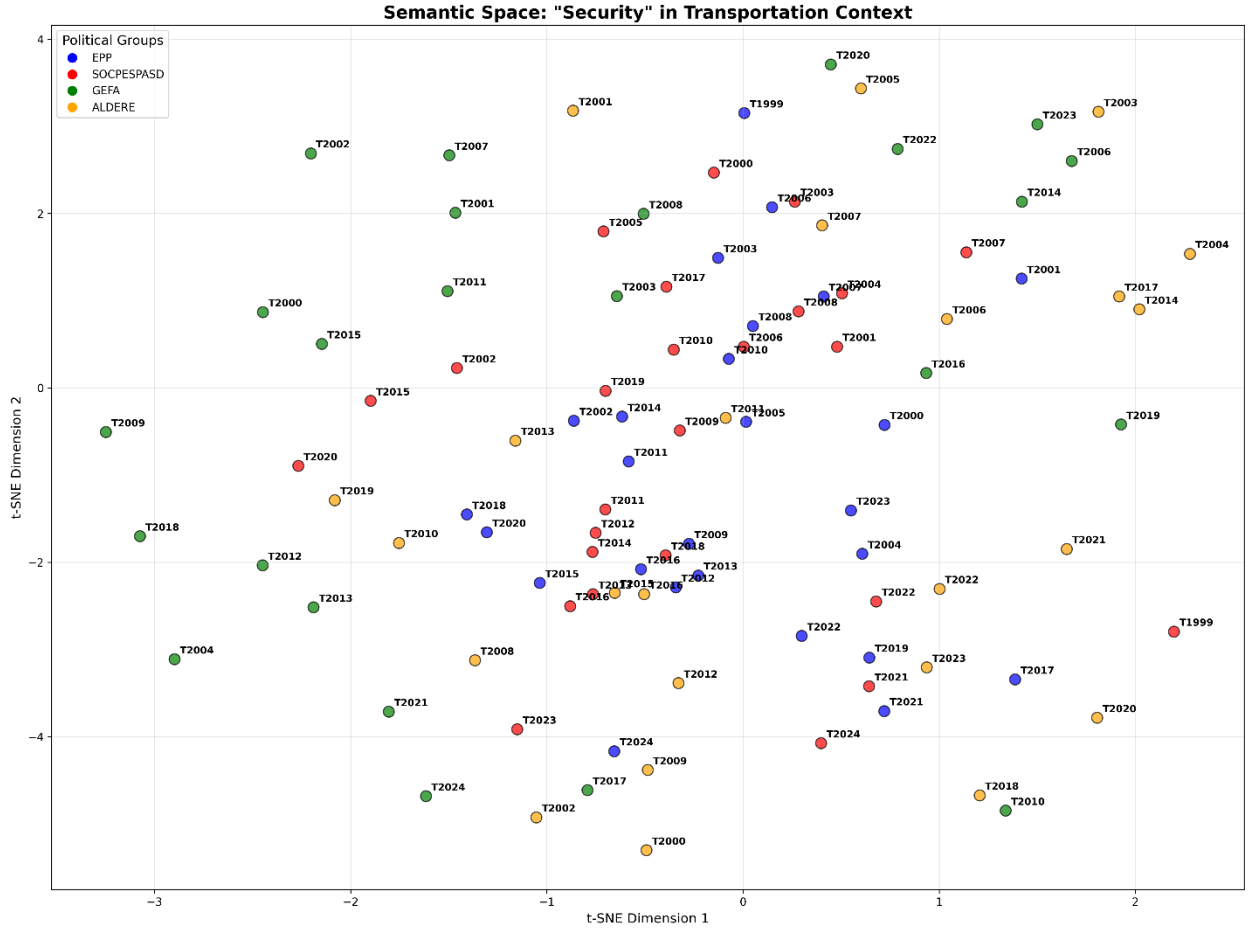
Relative polarization concentrates on GEFA vs GUENGL/NAT pairs, with late-period upticks.

5.7 Trade



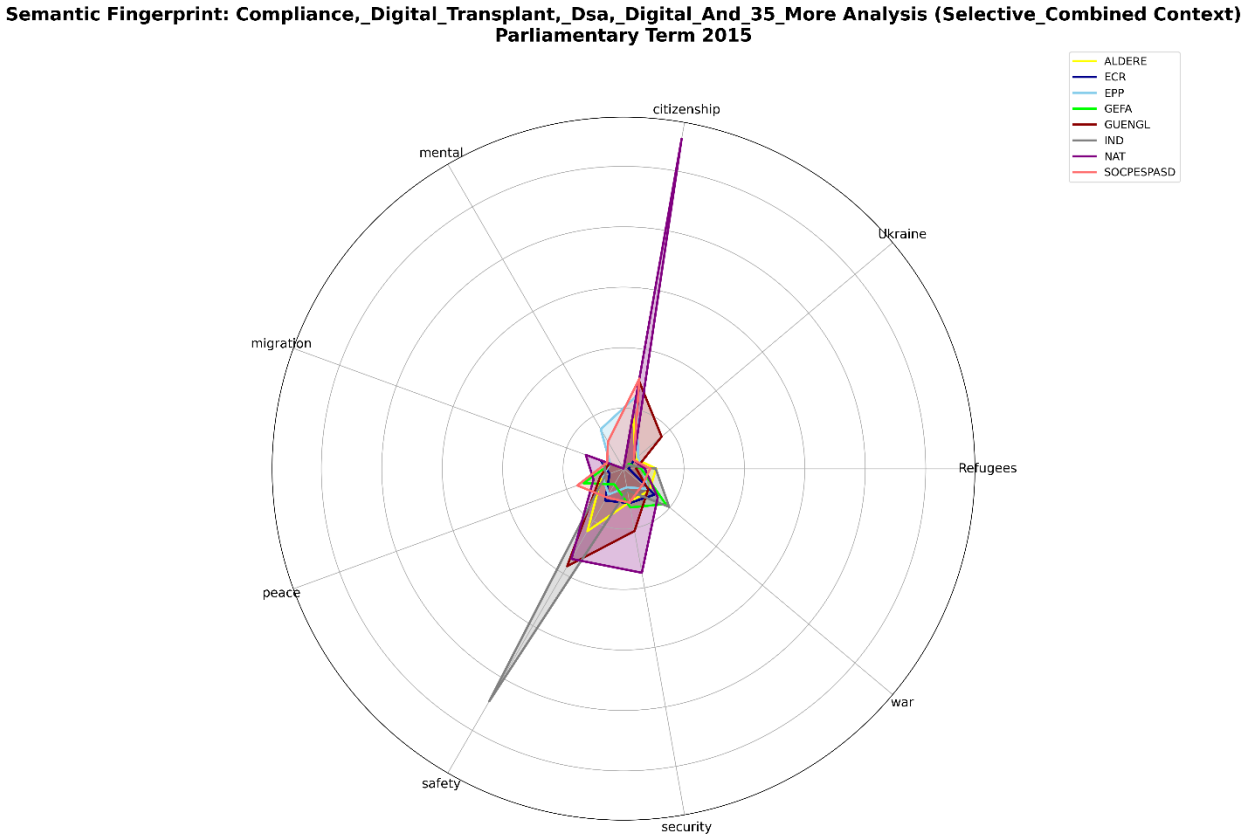
A large Security-coded block (sanctions, resilience, autonomy) sits opposite a coherent Immigration Island (visas/border implications), with a smaller Trade-labelled cluster (WTO/rules/market access). The multi-pole layout matches the event-driven ups/downs in the trade timelines.

5.8 Transportation

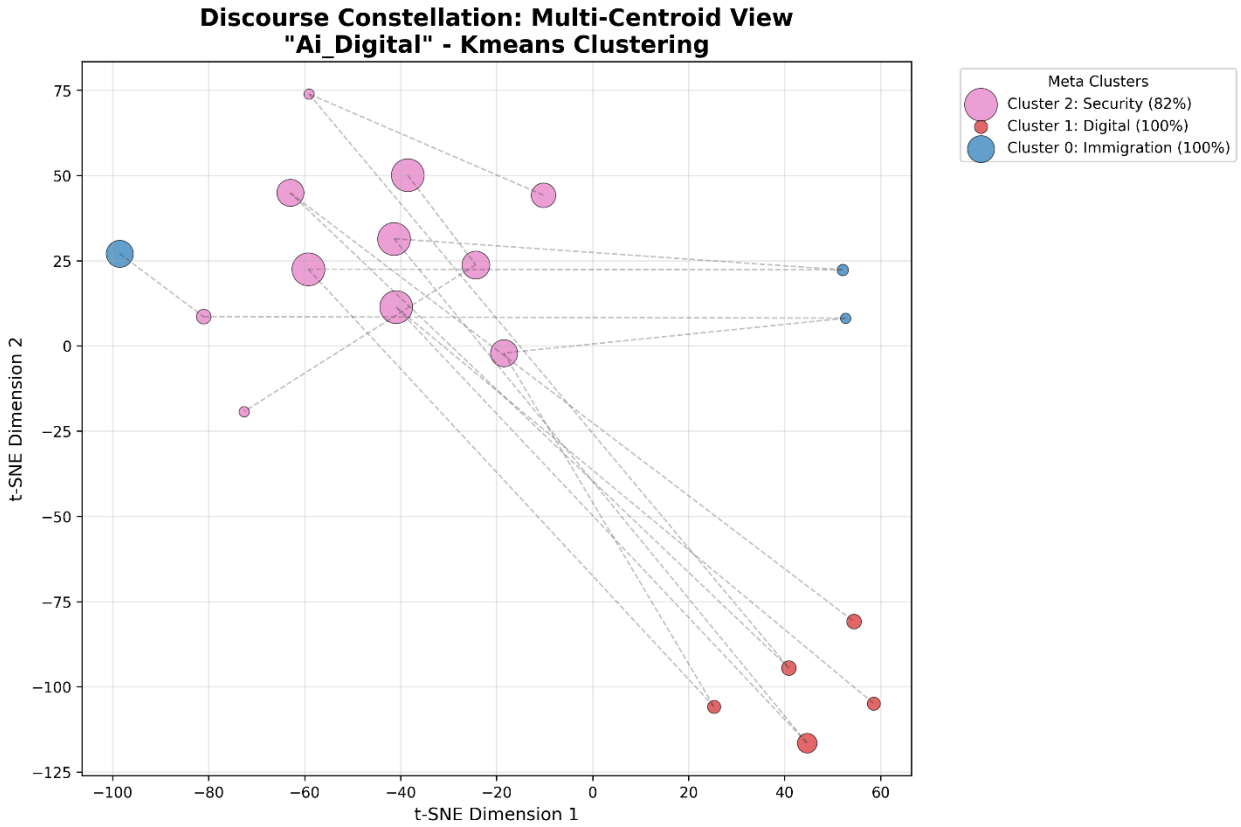


In transportation debates, parties cluster even more tightly, forming a dense ridge of shared language. Colors are thoroughly intermixed, implying common frames (infrastructure protection, critical networks, passenger safety) across groups. A few year-specific outliers appear at the edges, consistent with brief agenda shocks, but the overall structure points to limited, system-wide differentiation, again suggesting modest average polarization with short-lived bursts.

5.9 Digital and Data Governance

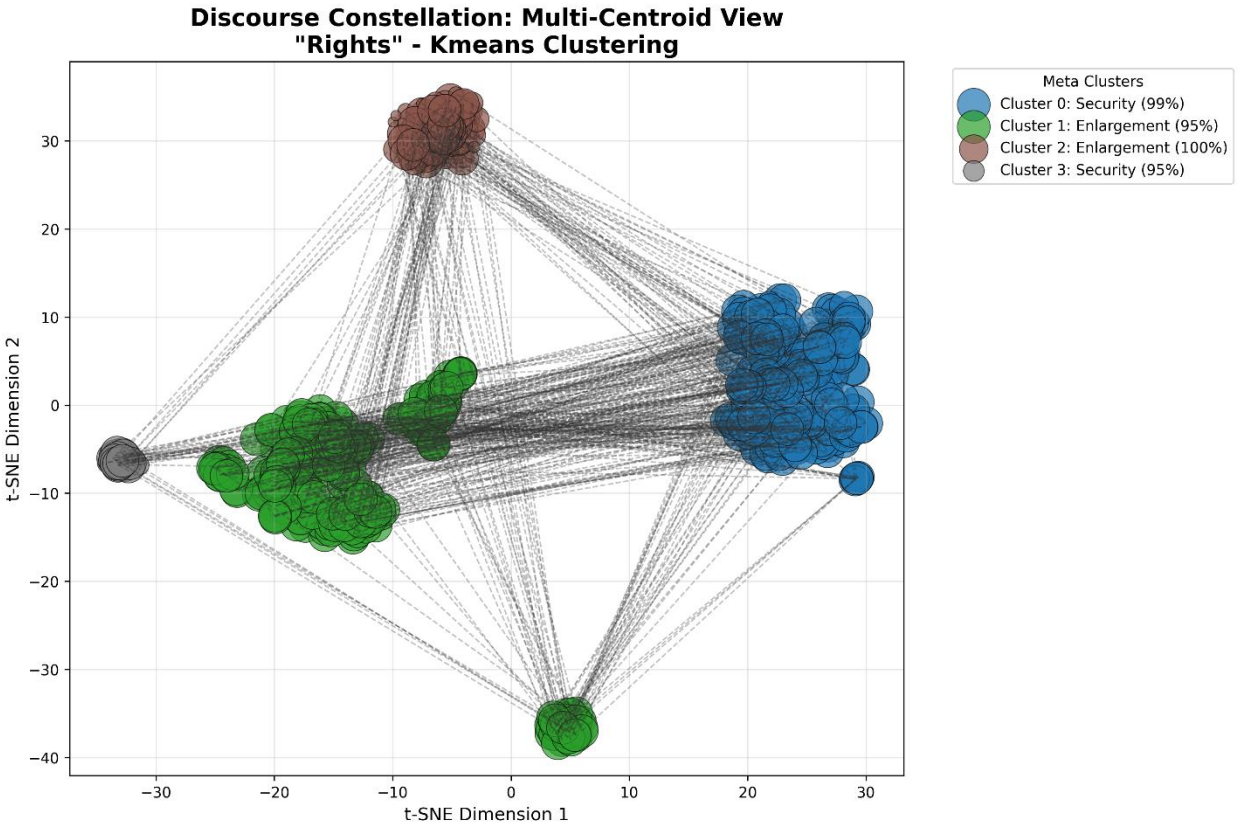


This figure shows the relative emphasis different groups place on core digital-governance terms in Term 2015. Spokes cover platform regulation, rights/citizenship, safety, and related policy vocabulary.

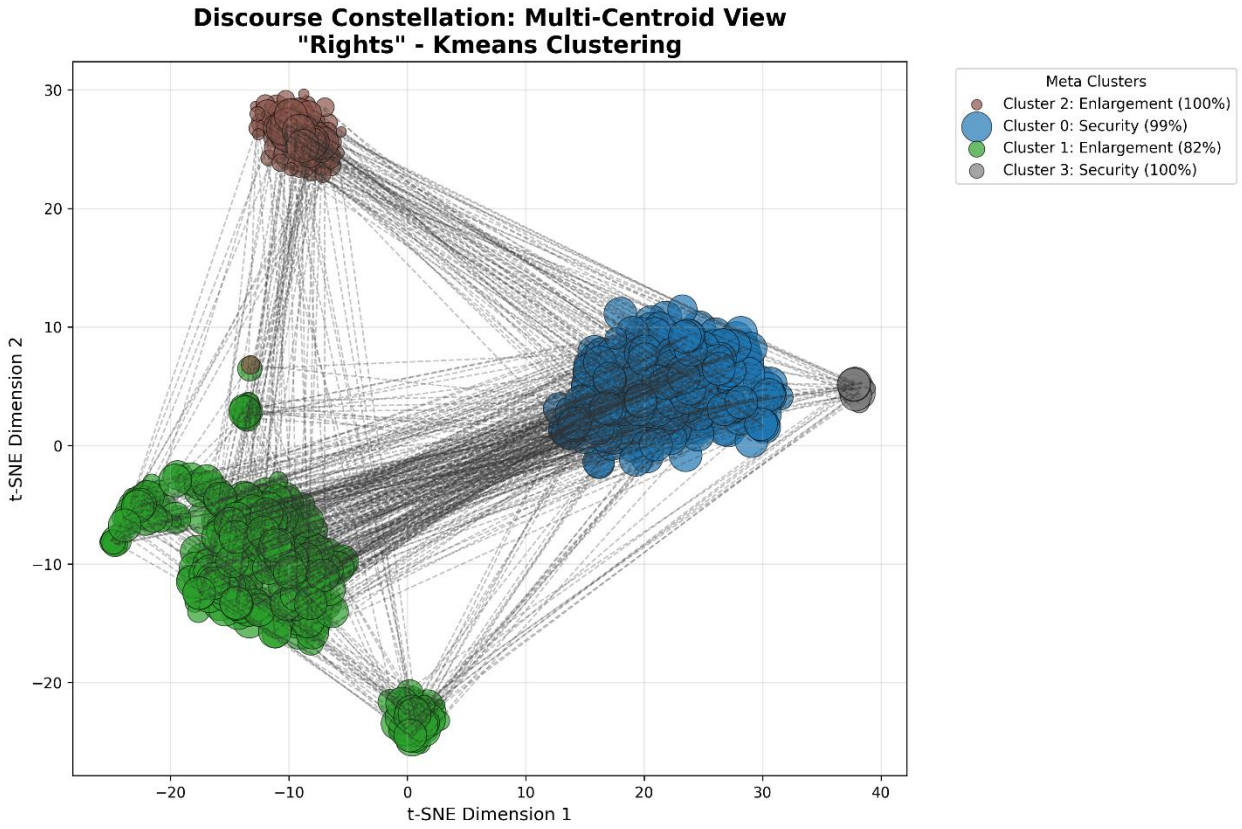


Discourse Constellation for the AI/Digital slice. Each small dot represents a party group×term point. Large bubbles are K-means centroids. Dashed lines link points to their centroid. Meta-label is Digital (100%), indicating a pure platform/rights/innovation framing with limited cross-leakage to other frames. Tight sub-clusters suggest broadly shared vocabulary around DSA/DMA/GDPR with modest between-group dispersion.

5.10 Enlargement

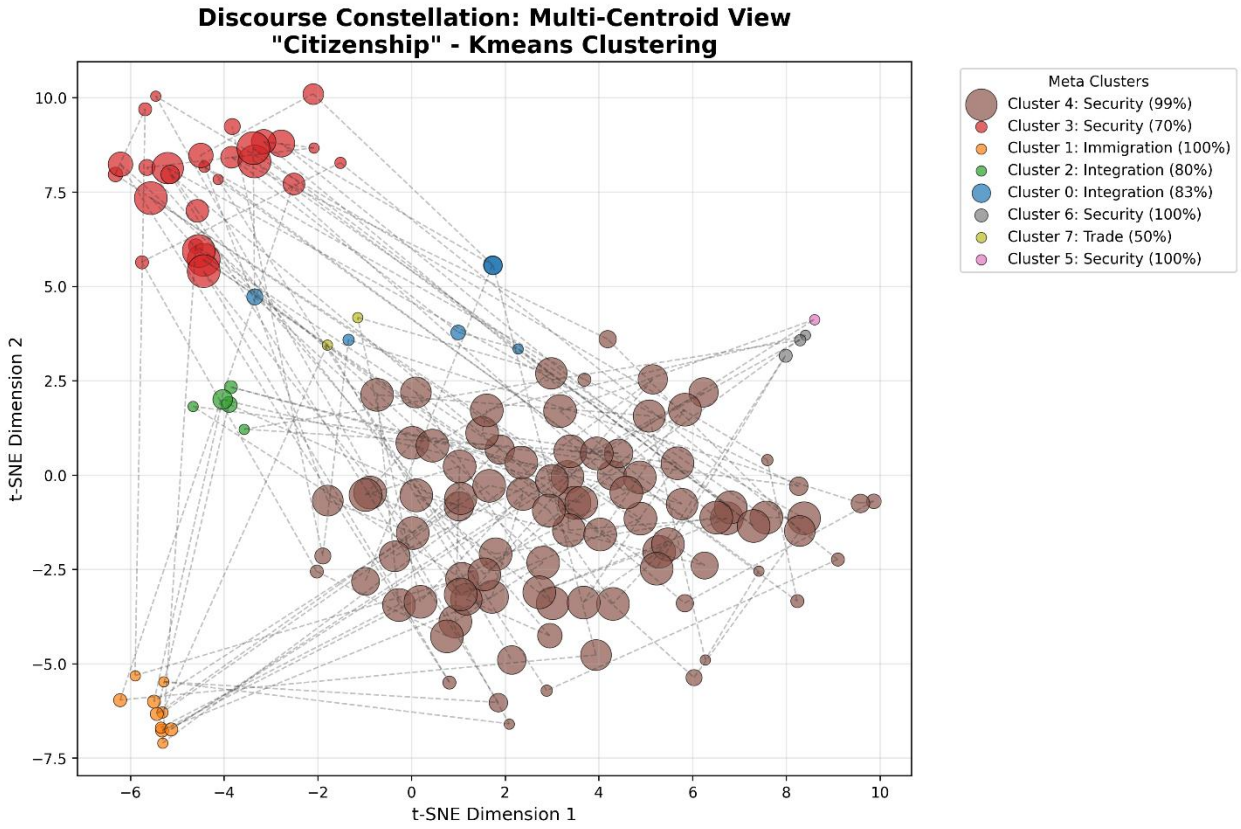


A more consolidated Enlargement reading of Rights (95%), with small Security satellites. When rights are anchored in accession conditionality, parties sound more alike; spikes arise when security intrudes

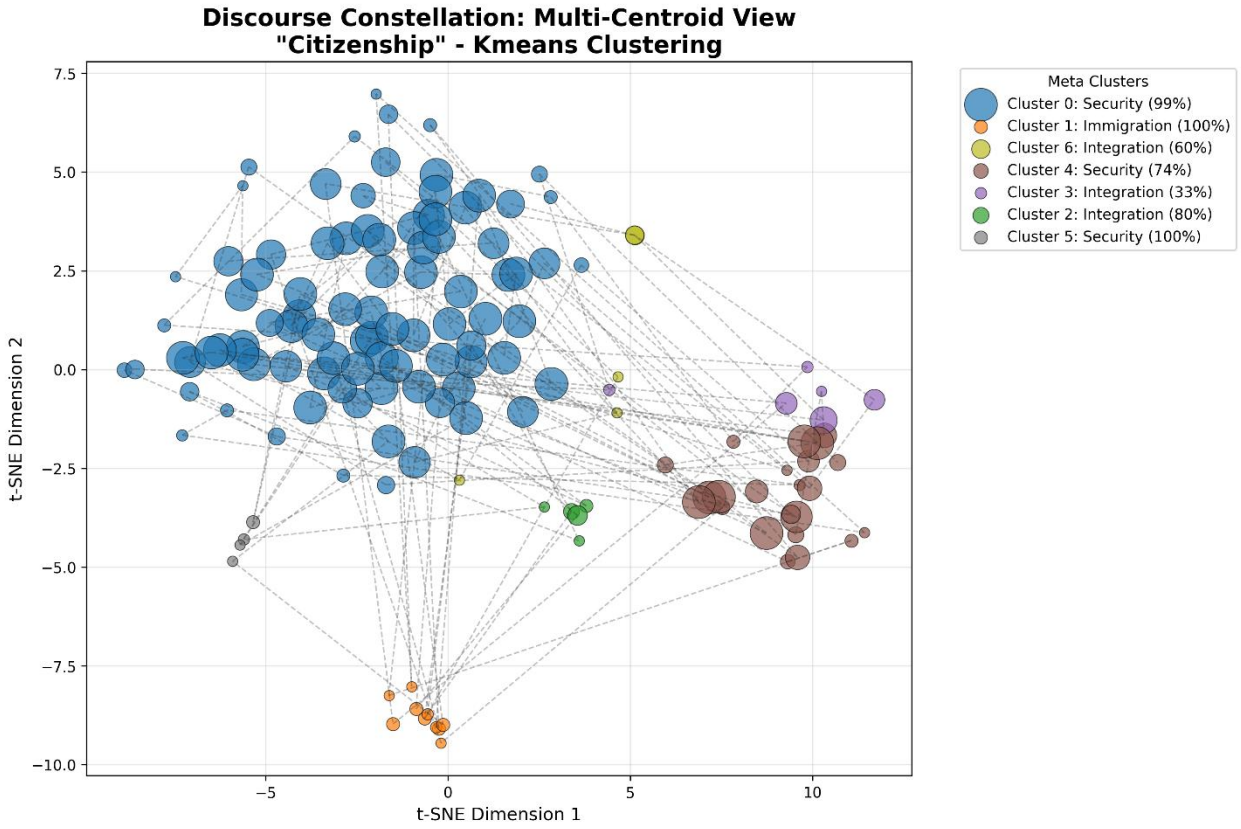


Rights talk in enlargement debates splits between Enlargement-coded clusters (accession chapters, rule-of-law criteria) and Security pockets (rights framed via safety/terrorism). This shows how conditionality and security logic co-exist.

5.11 Integration

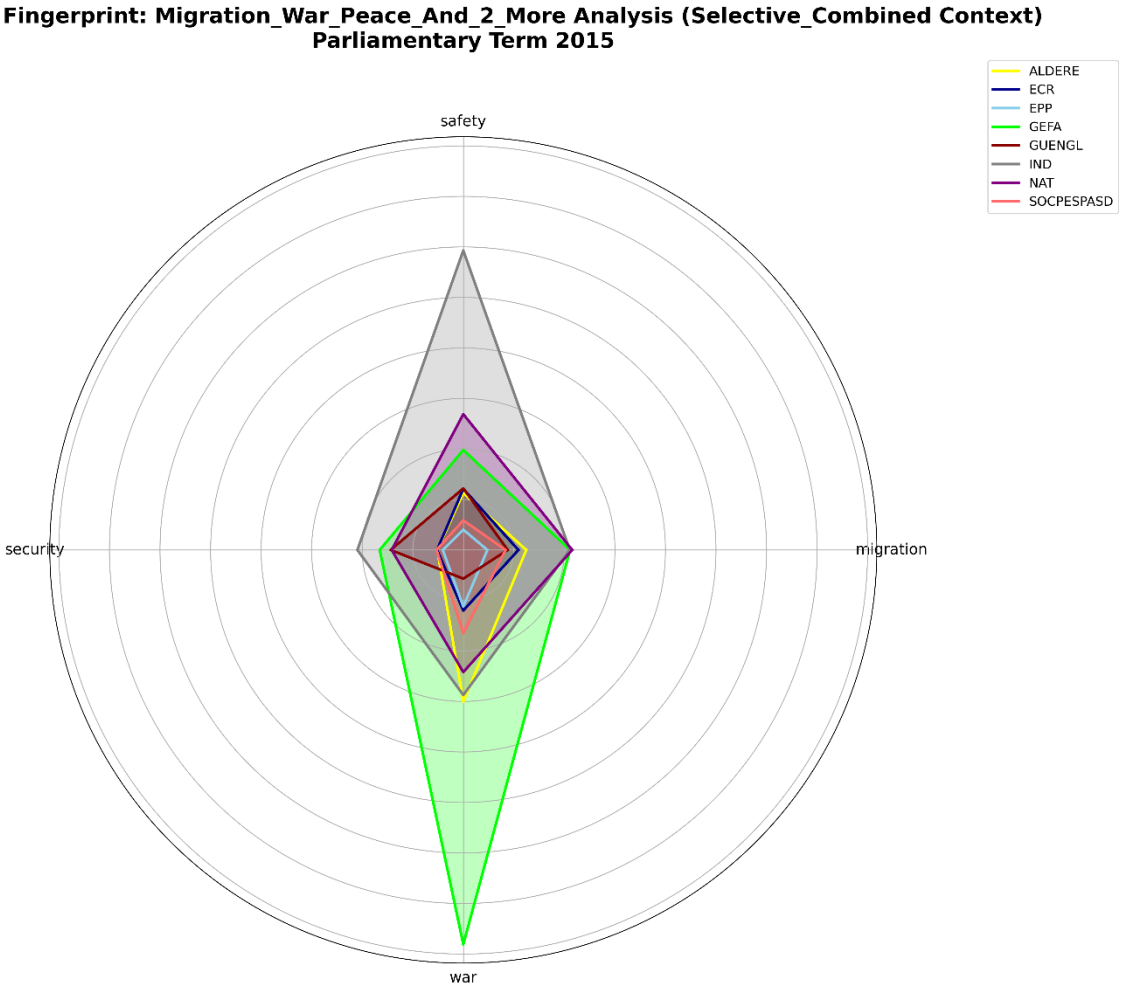


Citizenship constellation with a mixed profile (Security 70%). A large security cluster coexists with smaller Integration and Immigration pockets, indicating parallel emphases on rights/cohesion and control.

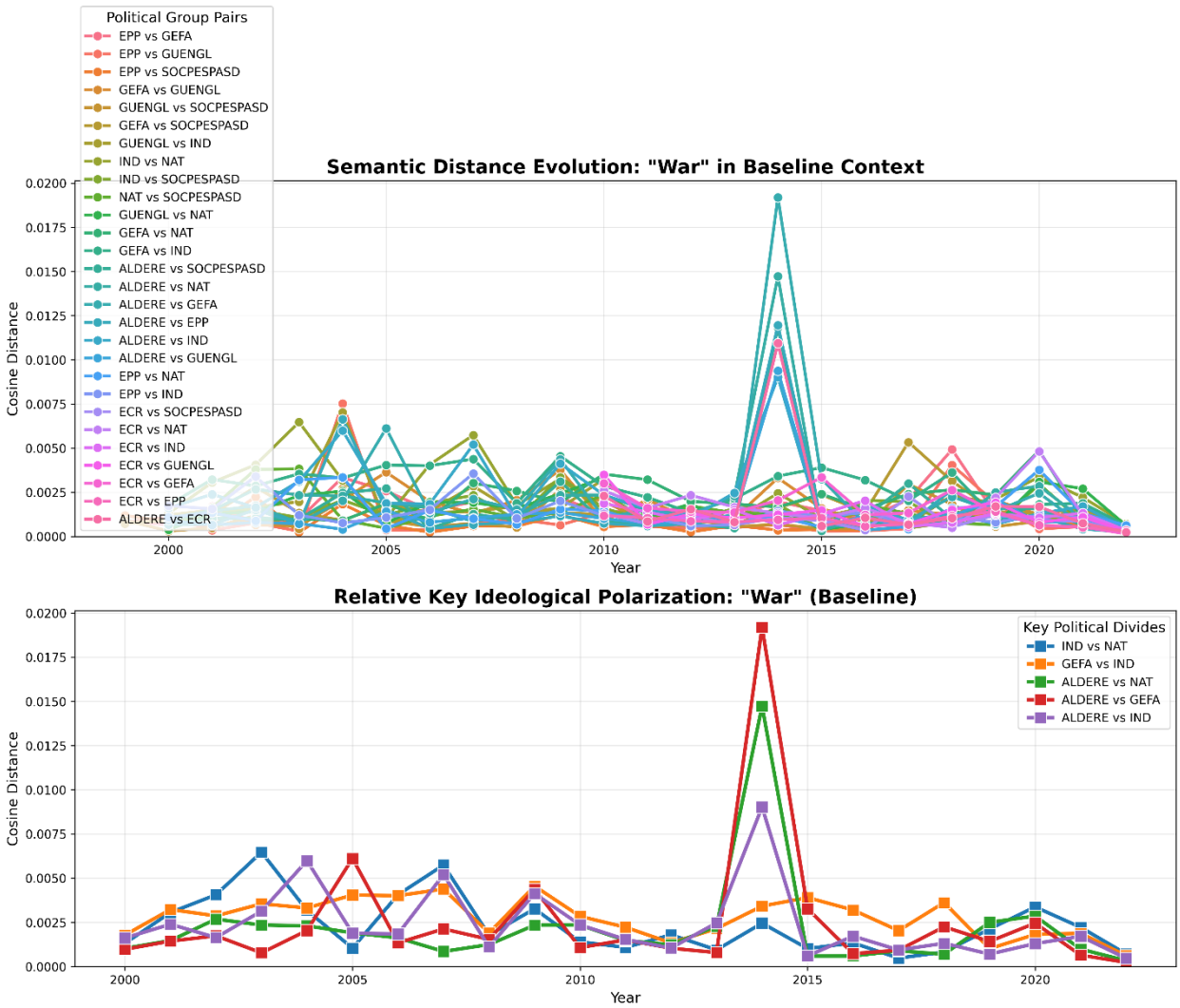


Slightly stronger security dominance (74%), but the Integration cluster remains structurally separate. Parties bridge the two frames unevenly, which is useful for explaining pairwise gaps in the timeline

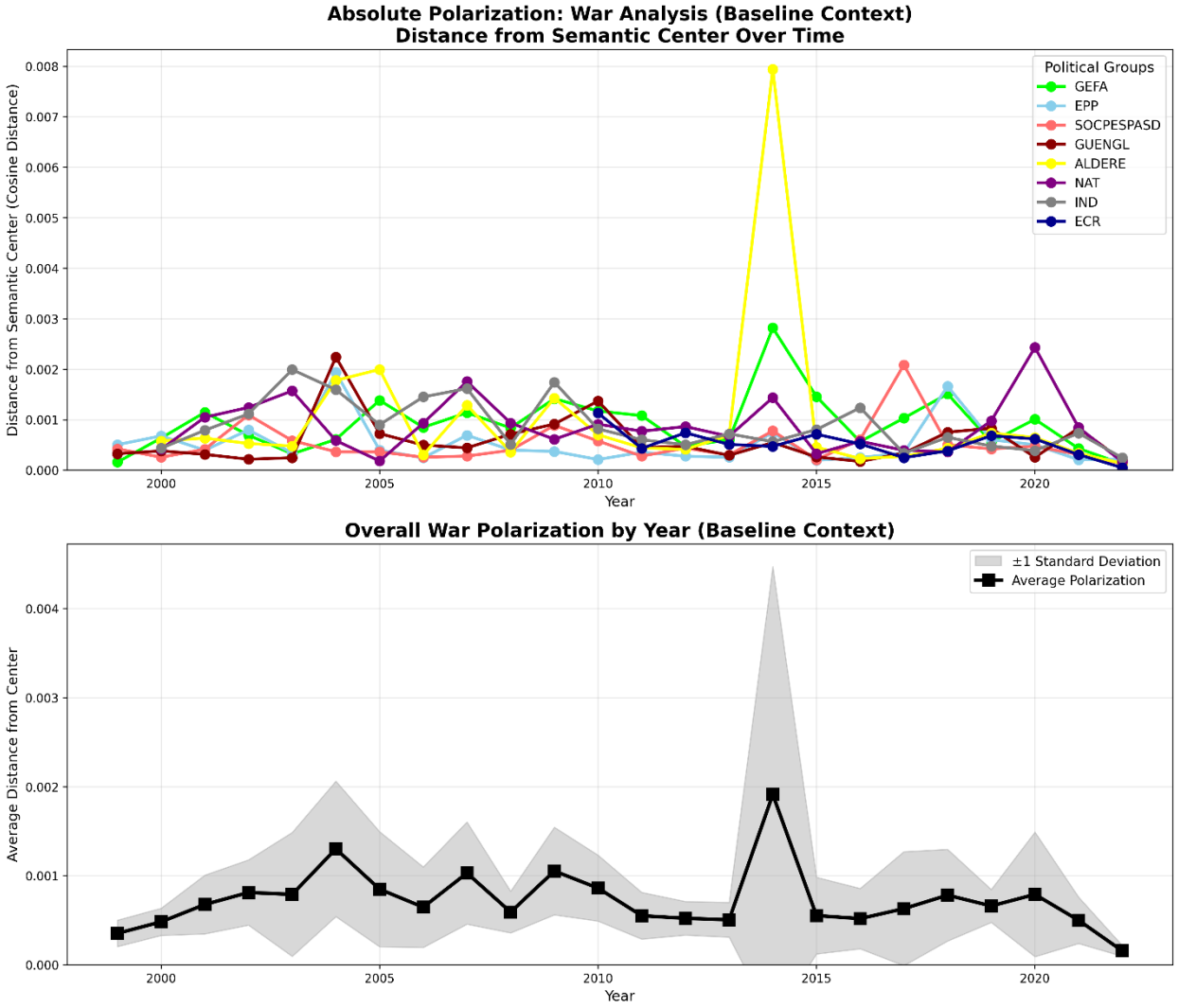
5.12 War



This radar shows how war-related language co-occurs with migration, safety, and security terms in Term 2015. It helps visualize which groups tie wartime framing more closely to migration/safety versus general peace/security rhetoric.



This figure tracks the cosine distance between party discourses over time. Polarization is generally low-to-moderate but shows a sharp, short-lived spike around the mid-2010s, indicating a temporary widening of inter-party frames on “War.” After the spike, distances settle back toward longer-run levels.



Absolute polarization around “War,” 1998–2024 (baseline). Party distances (top) and system average with ±1 SD (bottom). The system is generally centered but shows sharp, short-lived pulses (e.g., mid-2010s), after which the mean re-converges.

5.13 Cross-context takeaway

Across all policy areas, the parties share a common core when they say “security,” and the differences are about how they frame it. The biggest gaps show up mid-period for immigration and more recently for environment and economy. Hard security shows the most convergence over time. Remember the scale, distances are small (in the thousandths), so we’re looking at subtle but meaningful shifts in emphasis, not completely different meanings. Reading the maps and timelines together helps, maps reveal who sits with whom and timelines show exactly when those relationships tighten or loosen.

6. ROBUSTNESS, CAVEATS, AND INTERPRETABILITY

6.1 How to read the maps

I use t-SNE maps to give a quick picture of how party groups talk about a theme. On these maps, points that sit close together usually mean similar language and points far apart suggest different languages. However, t-SNE is a visual aid, not a ruler [6] :

- The two axes have no fixed meaning. I should not compare absolute positions across different maps as if the scales were the same.
- Small layout changes can happen if I change the random seed or the perplexity. That's normal for t-SNE

Because of this, whenever I want to compare precisely, I do not rely on the map alone. I go back to the polarization timelines (cosine distances) to make the quantitative call. My rule of thumb is to use maps for intuition and use timelines for numbers. When both tell the same story, clusters on the map and higher distances on the timeline, I have stronger evidence.

6.2 Data coverage and small cells

Not all-party groups talk about every theme in every period. If a group has only a handful of sentences on a theme each year or term, any average I compute would be unstable and could distort the results. To avoid this:

- I set a minimum sentence threshold for each (group, time, theme) cell.
- If a cell falls below that threshold, I exclude it from that analysis.
- I log into every exclusion with the reason and keep a summary of how many cells passed or failed.
  This way, I protect the analysis from noise caused by thin data and I give myself (and readers) a clear audit trail. If I later change the threshold or the keyword list, I can rerun the pipeline and immediately see how the coverage changes.

6.3 Clustering choices and baselines

Sometimes a party group talks about the same theme in several distinct ways (for example, "security" in a border sense and a cyber sense). To capture this, I offer two options:

- Average pooling (baseline). I take all sentence vectors for a (group, time, theme) cell and compute one average vector. This is fast, stable, and often good enough.
- KMeans with Dynamic K. I try several values of K and pick the smallest K that reaches a reasonable silhouette score. [8] This lets me separate sub-topics inside the same theme and, if I want, analyze the centroids as distinct focuses.
  In practice, I default to average pooling and switch on clustering when I suspect clear sub-themes or when the average looks like it's hiding multiple centers. Even when I use clustering, I can always collapse the clusters back into a single weighted average so that downstream plots remain comparable. This keeps the method flexible without sacrificing consistency.

7. REPRODUCIBILITY GUIDE

7.1 Single-context run

Goal: analyze one keyword in one theme, across parliamentary terms.

```
python run_poc_pipeline.py \
  --temporal term \
  --keywords security \
  --themes environment \
  --model Roberta
```

What does this do, step by step:

- Phase 1 – Data prep: splits speeches into sentences, filters to the environment theme using the security keyword and its variants and saves the filtered sentences to run_*/data/.
- Phase 2 – Embeddings: turns those sentences into vectors using the Roberta model; saves to run_*/data/.
- Phase 3 – Aggregation: builds one vector per party group × term (average pooling by default); saves to run_*/data/.
- Phase 4 – Analysis & plots: computes pairwise distances over time and draws:
    - run_*/visualizations/poc_polarization_over_time_environment.png
    - run_*/visualizations/poc_semantic_space_tsne_environment.png

How I read the outputs:

- The timeline shows how far apart the party groups are (cosine distance) for security in the environment across terms.
- The t-SNE map gives a quick visual of who is close and who is far in that theme.

Tip: switch to yearly analysis by changing --temporal:

--temporal year.

7.2 All contexts at once

Goal: run the same keyword across every theme in one go.

```
python run_poc_pipeline.py \
  --temporal term \
  --keywords security \
  --themes all \
  --model Roberta
```

What do you get:

- A complete set of figures, one pair per theme, for example:
  - poc_polarization_over_time_{economy|energy|environment|health|immigration|security|trade|transportation|defence|digital|enlargement|integration|war}.png
  - and the matching poc_semantic_space_tsne_<theme>.png files.
- All of these live under run_*/visualizations/, with the filtered data and embeddings under run_*/data/, and a detailed log under run_*/logs/.

Good to know:

- The run is modular. If you rerun with the same settings, cached files are reused, so later runs are faster.
- If a (group, term) cell is too small (not enough sentences), it's skipped, and the reason is written in the logs. This keeps results stable.

7.3 Grouped-keyword analysis

Goal: study a bundle of related words as one concept (for example, "security+defence" together).

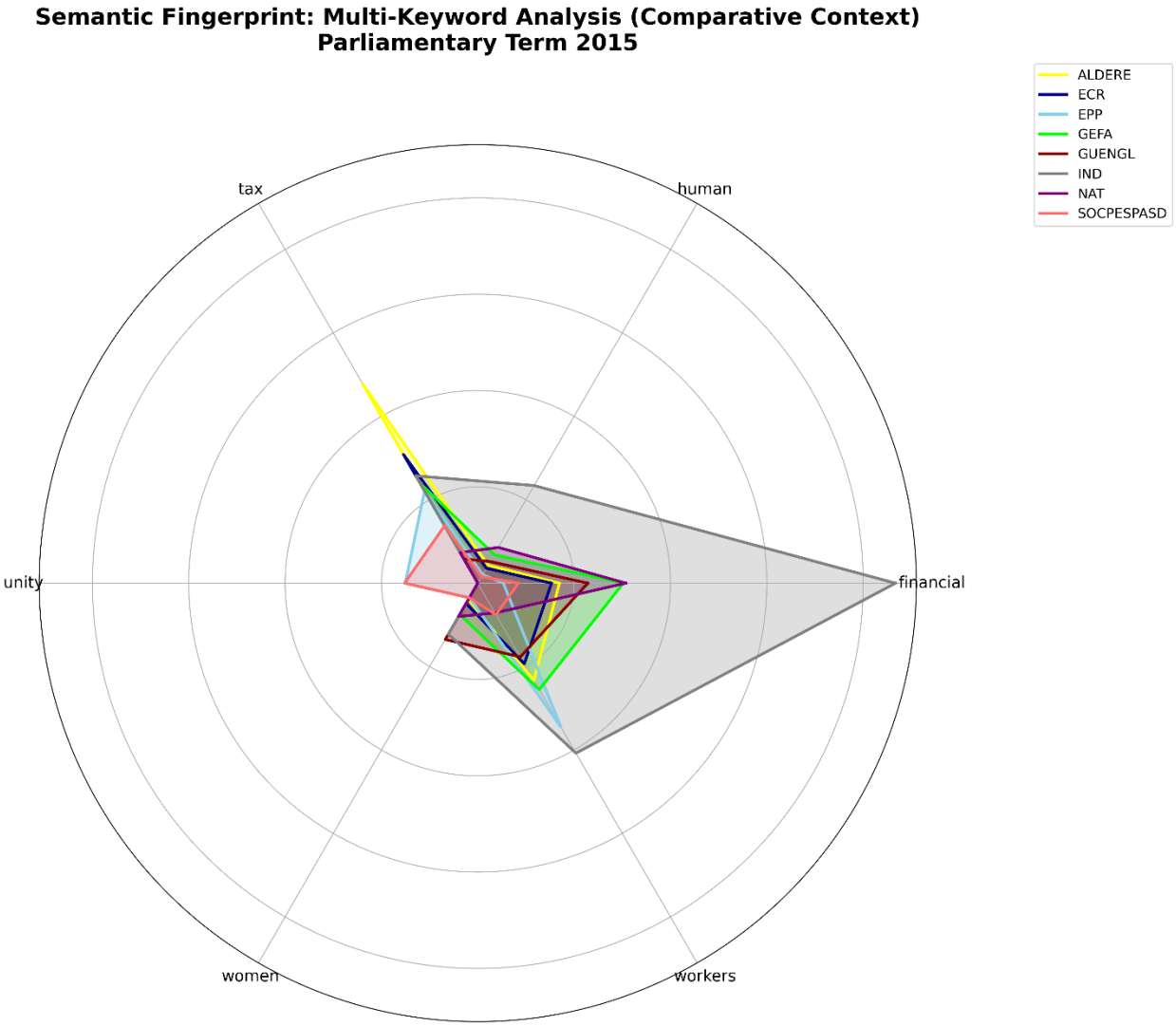
```
python run_poc_pipeline.py \
  --temporal term \
  --keyword-groups "security+defence" \
  --themes security \
  --model roberta
```

What happens:

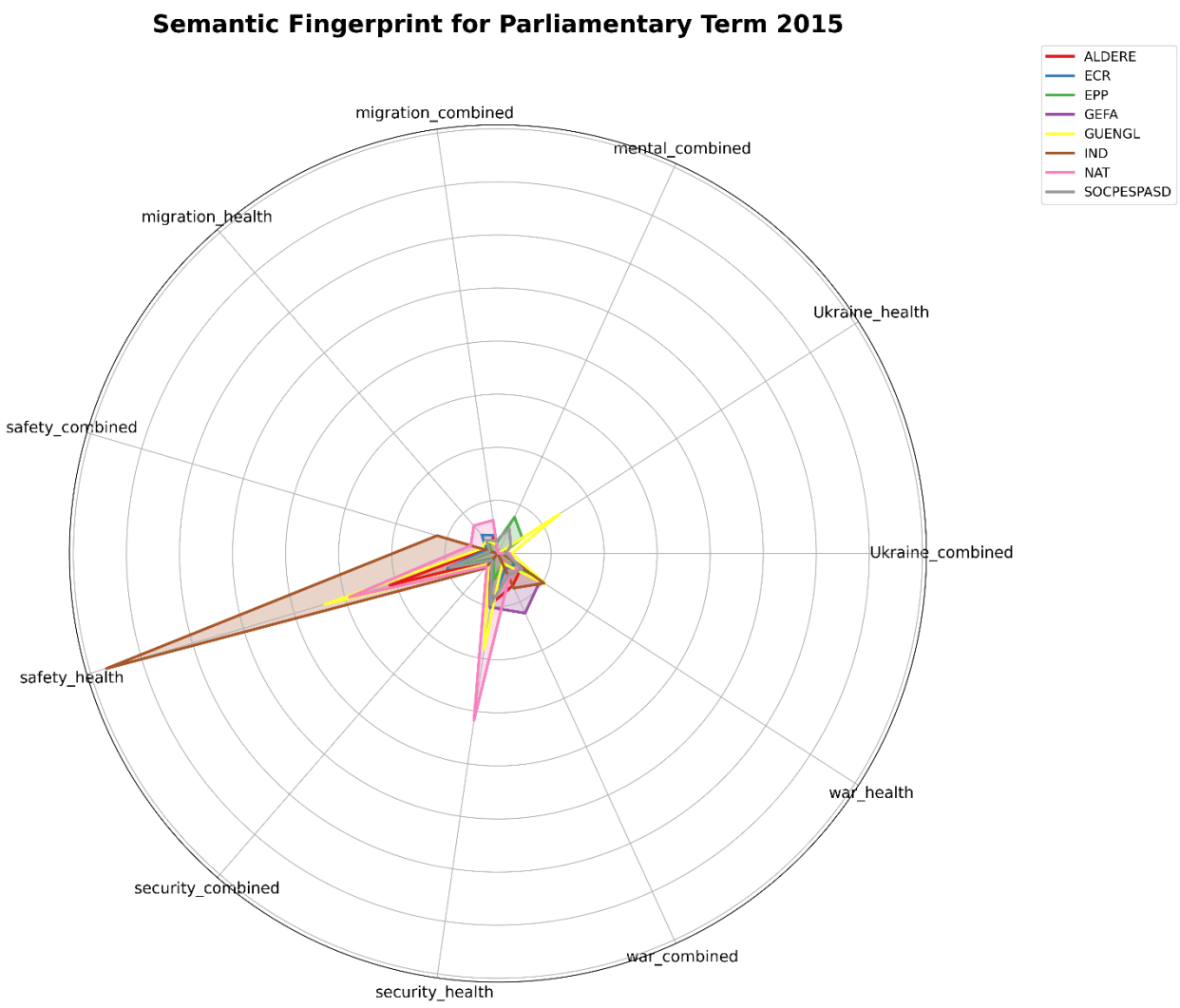
- Phase 1 collects sentences that match any of the words in the group (here: security OR defence), including their variants (stems/lemmas).
- Phases 2–3 create embeddings and group-time summaries as usual, but now for the combined concept.
- Phase 4 writes outputs with clear names, e.g.:
  - poc_polarization_over_time_security__security+defence.png
  - poc_semantic_space_tsne_security__ security+defence.png

When I use this:

- When a theme is naturally multi-word (e.g., digital+platforms+AI), grouping improves coverage and stability without mixing in unrelated text.



This example illustrates the grouped-keyword option (e.g., “security+defence” or broader social-economic bundles). The plot shows normalized salience for each keyword within the grouped concept, by party group.



This radar demonstrates paired-concept fingerprints (e.g., “security+health”). It visualizes how often and how strongly party groups use a focal idea together with a second lens, offering a compact view of cross-theme co-framing within one term.

8. RELATED METHODS

8.1 What others usually do and why it wasn't enough for me

A lot of political text work uses bag-of-words or dictionary lists. [7] These methods count words but ignore order and context. They are fast and easy to explain, but they miss crucial differences like "border security" vs "energy security." Older static word vectors (for example, Word2Vec or Glove) go a step further by mapping each word to numbers, but they still give one vector per word**,** no matter where it appears. That means they struggle with polysemy (one word, many meanings) and with strategic framing in political speech.

8.2 Why I use contextual transformers

I use contextual transformer encoders (RoBERTa-style models) because they read the whole sentence before deciding how to represent a word. In practice, the vector for "security" changes depending on whether the sentence talks about borders, energy, health, or digital platforms. This is exactly what I need to compare how party groups frame the same idea in different policy areas

8.3 Two levels of meaning I extract

I work at two complementary levels:

- Sentence level. I turn the whole sentence into a single vector. [4] This gives me a broad, stable signal of how a group talks about the theme overall.
- Keyword-in-context level. [2] [3] When I care about a specific word (like security), I locate that word in the model's tokens and pool the hidden states for those tokens. This produces a vector for that word in that sentence, preserving fine-grained meaning. If tokenization splits the word into pieces, I use safe fallbacks, so I always get a useful representation.

Together, these two levels let me see both the big picture and the nuances.

8.4 How this compares to alternatives I considered

- Bag-of-words / TF-IDF. Great for speed and transparency, but they ignore word order and context and they often confuse distinct senses of the same term.
- Dictionaries/lexicons. Useful for targeted searches, but they can be brittle across languages and miss paraphrases.

- Topic models (e.g., LDA). It's good for discovering themes, but topics can mix frames and don't directly tell me how one word is used differently across contexts.
- Static embeddings (Word2Vec/Glove) or doc2vec. Capture some similarity, but one vector per word can't handle contextual shifts that are central to political framing.

Contextual transformers solve the exact problem I face, which is that the same word carries different meanings depending on policy area and moment in time.

8.5 Practical trade-offs and checks

Contextual models are heavier than older methods, so I:

- Batch sentences and use the GPU when available.
- Keep the model pluggable (e.g., RoBERTa for quality, DistilRoBERTa for speed).
- Add simple sanity checks (nearest-neighbor inspection, stability across random seeds) to make sure vectors behave the way I expect.

If I ever need a lighter baseline, I can still fall back to TF-IDF or static embeddings for rough comparisons, but the main results use contextual models.

8.6 How I wired it into the pipeline

In code, I expose clean hooks to:

- Encode sentences into vectors.
- Extract token spans for a keyword and pool the hidden states.
- Swap models with a --model flag without touching downstream steps.

These hooks make the method reusable and easy to extend, for example, to add a multilingual model, test a newer encoder, or experiment with different pooling strategies.

9. FILE INVENTORY

9.1 poc_phase1_data_prep.py

What it does:
This is my data entry point. I load the European Parliament speeches (RDS file), discover which political groups are present, and split speeches into sentences. I also clean the text and (when useful) add lemmas/stems so I can catch different forms of the same word (e.g., security, securing).

How do I pick the right sentences?
I apply my THEMATIC_FILTERS (keyword lists per theme). I expand each keyword to include its variants (stems/lemmas) and keep only the sentences that match the theme.

Quality control:
If a party group has too few sentences in each period (year/term), I drop that cell to avoid noisy averages. Every exclusion is logged.

Outputs:
I save one or more files like poc_<keyword-or-group>_<theme>_sentences.pkl under the run's data/ folder. These are the inputs for the next phase.

9.2 poc_phase2_embeddings.py

What it does:
Here I set up the transformer model (e.g., RoBERTa or DistilRoBERTa) and turn sentences into embeddings (vectors). I also provide a helper called extract_keyword_embedding to grab the vector for a specific word inside a sentence (keyword-in-context), with safe fallbacks if tokenization is tricky.

Why it matters:
This is where meaning becomes numbers. Downstream steps rely on these vectors to compare party groups.

Outputs:
Files like poc_*_<model>_embeddings.pkl (saved under data/) that hold the sentence or keyword-in-context vectors ready for aggregation.

9.3 poc_phase3_aggregation.py

What it does:
I take the Phase-2 embeddings and summarize them per party group × time. I offer two modes:

- Average: one stable vector per group and time (fast, reliable baseline).
- KMeans (Dynamic K): when I expect multiple sub-topics inside a theme, I pick K using a silhouette quality check and keep centroids and labels.

Outputs:

- …_avg_embeddings.pkl for average pooling, or
- …_clustered_data.pkl for clustered summaries.
  Both live under the run's data/ folder and feed the plots in Phase 4.

9.4 poc_phase3_analysis.py

What it does:
This is a helper/utility module. It includes convenience loaders, small summaries, and some plotting glue (colors, legends) that I reuse across figures. Think of it as my toolbox for making analysis steps concise and consistent.

9.5 poc_phase4_visualization.py

What it does:
I load the group-time vectors and compute the metrics I show in the thesis:

- Pairwise cosine distances (my measure of relative polarization) over time.
- t-SNE maps (simple 2-D views of the semantic space) per theme.
- Optional theme concept vectors (short prompts embedded and averaged) to support comparative panels.

Outputs:
Saved figures under visualizations/, including:

- poc_polarization_over_time_<theme>.png
- poc_semantic_space_tsne_<theme>.png
  These are the main pictures I discuss in the results chapter.

9.6 eu_discourse_analysis.py

What it does:
Project-level helpers that Phase 4 imports. It centralizes analysis routines, so I don't repeat code: loading/saving patterns, small math helpers, and consistent naming rules for outputs. It keeps the pipeline organized and easier to maintain.

9.7 plotting_utils.py

What it does:
One place for shared visuals: party color maps, font sizes, layout defaults, safe figure saving (with overwritten protection), and small helpers for legends and labels. This ensures that all plots look consistent across themes and runs.

9.8 run_poc_pipeline.py

What it does:
This is the orchestrator I run from the command line. It:

- accepts flags like --temporal term|year, --themes …, --keywords … or --keyword-groups …, and --model …,
- runs all phases in order,
- creates the timestamped run folder with clean sub-folders (data/, visualizations/, logs/, and metadata),
- supports resume/list (if enabled in your setup) so I can continue an interrupted run or inspect prior runs,
- prints a short success report at the end.

9.9 poc_logger.py and tee_logger.py

What they do:
These files handle structured logging. They print clear phase banners (e.g., PHASE 1: FOCUSED DATA PREPARATION …), show dataset summaries, and capture stdout/stderr to log files. If anything goes wrong, I can open the logs and see exactly where and why.

9.10 Theme lists & notes (PDF)

What it is:
A short document with the curated keyword lists for each theme and notes on how to read the visuals (for example, “t-SNE is illustrative”), plus any consolidated category lists I maintain.

Why it matters:
It keeps my assumptions visible, which words define each theme, which variants I include, and the caveats I expect readers to remember. I treat it like a small data dictionary for themes.

10. HOW TO READ THE FIGURES (QUICK GUIDE)

10.1 Polarization timelines

Files: poc_polarization_over_time_<context>.png

These charts show how far apart party groups are when they talk about the same theme over terms or years. Each line is a pair of groups; the y-axis is the cosine distance between their language (higher = more different, lower = more similar). The numbers are usually small (often in the thousandths), so even steady, small gaps can be meaningful if they persist over time.

How I read them:

- I look for trends: Do lines rise (groups drifting apart) or fall (groups getting closer)?
- I look for turning points: Do lines spike or drop around key periods?
- I check which pairs are consistently closest (often natural allies) and which pairs are consistently far (stable disagreement).
- If a line is missing in a period, it usually means that the (group, time, theme) cell didn't have enough sentences and I filtered it out for reliability.

10.2 t-SNE semantic map

Files: poc_semantic_space_tsne_<context>.png

These are simple 2-D maps of the "semantic space." Each point is a party group in each period. Nearby points use language in similar ways, far points sound different. Colors stay consistent by party, so you can follow a group across periods.

How I read them:

- I look for clusters (groups that sit together) and outliers (groups off by themselves).
- I follow a party's path across periods (e.g., T7 → T8 → T9) to see if it drifts toward or away from others.
- I use these maps to spot patterns fast, then I go back to the timelines to confirm the strength of those patterns.

10.3 Comparative panels

Sometimes I place contexts next to each other (for example, environment vs economy in the most recent term) to show where rhetoric diverges most right now. These panels can be two timelines, two t-SNE maps, or one of each.

How I read them:

- First, I check that the y-axes on timelines use comparable scales. If the scales differ, I focus on the shape (rising/falling) rather than the raw height.
- Then, I compare the most recent period (e.g., T9) across the chosen contexts: Where are the largest current gaps? Which parties moved the most?
- I use the maps to see who sits with whom, and the timelines to see when those relationships tightened or loosened.

11. CONCLUSION

In this thesis, I built a clear end-to-end pipeline to track how European Parliament party groups talk about the same idea across policy areas and over time. I work at the sentence level, turn meaning into vectors with a modern language model, and then measure how close or far groups are. The process is fully reproducible, like each step writes files, every run is logged and figures use consistent names.

Methodologically, this adds to classic tools like roll-call analysis and topic models. Roll-calls show how groups vote; topic models show major themes. My method shows how ideas are framed and measures semantic distance in context. So, I can tell when two groups use similar language about “security” in energy but diverge on “security” in immigration. This fills the gap between counting votes and discovering topics.

Results show a shared core, like most of the time, parties use language that is not wildly different. Distances are small, meaning differences are about emphasis and framing. Still, there are clear splits, strongest mid-period in immigration. I also find recent divergence in environment and economy, matching tensions over costs, transition speed, competitiveness, and social impact.

I also see convergence in hard-security language, like early gaps shrinking, reopening only slightly later. In trade, transport, and digital, patterns move with events, spikes and dips around shocks and major files, then settle.

I interpret carefully. t-SNE helps with intuition, but I rely on distance timelines for precise claims. I filter thin data and keep detailed logs for all inclusions and exclusions.

Practically, the pipeline is modular and reusable, like I can swap models, add themes, run by term or year, and resume easily. Consistent outputs let me compare runs and build on them. Overall, I show we can track meaning in context, beyond word counts or votes, in a transparent, repeatable, readable way, like a common core with periodic, theme-specific splits and narrowing gaps in hard security.

12. TABLE OF TERMINOLOGY

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Semantic polarization	Σημασιολογική πόλωση
Contextual embeddings	Ενσωματώσεις σε συμφραζόμενα
Sentence embedding	Ενσωμάτωση πρότασης
Keyword-in-context	Λέξη-κλειδί στα συμφραζόμενα
Transformer model	Μοντέλο transformer (μετασχηματιστή)
Tokenization	Τοκενοποίηση (κατάτμηση σε υπομονάδες)
Vector dding vector	Διάνυσμα / διανυσματική αναπαράσταση
Cosine distance	Απόσταση συνημιτόνου
Dimensionality reduction	Μείωση διαστάσεων
t-SNE (technique for DR)	t-SNE (τεχνική μείωσης διαστάσεων)
K-Means clustering	Ομαδοποίηση K-Means
Centroid	Κεντροειδές
Silhouette score	Δείκτης σιλουέτας
Word cloud	Νέφος λέξεων
Lexical signature	Λεξικό αποτύπωμα
Aggregation / pooling	Συνάθροιση / pooling
Pipeline	Pipeline (διαδικασία ροής)
Polarization timeline	Χρονογραμμή πόλωσης
Semantic map	Σημασιολογικός χάρτης
Distance-to-centroid	Απόσταση από κεντροειδές
Parliamentary term	Κοινοβουλευτική περίοδος
Party group	Πολιτική ομάδα

13. ABBREVIATIONS - ACRONYMS

EU	European Union
EP	European Parliament
EUPDCorp	Corpus of the EU Parliament Debates
BSc	Bachelor of Science
EPP	European People's Party
S&D	Progressive Alliance of Socialists and Democrats
Greens/EFA	Greens/European Free Alliance
GUE/NGL (GUENGL)	European United Left–Nordic Green Left
ECR	European Conservatives and Reformists
ALDE / RE	Alliance of Liberals and Democrats for Europe / Renew Europe
GDPR	General Data Protection Regulation
DSA	Digital Services Act
DMA	Digital Markets Act
WTO	World Trade Organization
t-SNE	t-distributed Stochastic Neighbor Embedding
TF-IDF	Term Frequency–Inverse Document Frequency
LDA	Latent Dirichlet Allocation
RDS	R Data Serialization
CSV	Comma-Separated Values
PNG	Portable Network Graphics
GPU	Graphics Processing Unit
NAACL-HLT	North American Chapter of the ACL: Human Language Technologies
ACL	Association for Computational Linguistics
EMNLP	Empirical Methods in Natural Language Processing
W3C	World Wide Web Consortium
ITU	International Telecommunication Union
IEEE	Institute of Electrical and Electronics Engineers
SOAP	Simple Object Access Protocol
ISO	International Organization for Standardization
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
A.M.	Αριθμός Μητρώου
S.N.	Student Number

14. BIBLIOGRAPHY

[1] M. Mochtak, “Corpus of the EU Parliament Debates (EUPDCorp), 1999–2024 (1.0),” Zenodo, dataset, 2025. doi: 10.5281/zenodo.15056399. [Online]. Available: https://doi.org/10.5281/zenodo.15056399

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proc. NAACL-HLT*, 2019. [Online]. Available: https://aclanthology.org/N19-1423/

[3] Y. Liu, M. Ott, N. Goyal, *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *arXiv* preprint arXiv:1907.11692, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[4] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proc. EMNLP*, 2019. [Online]. Available: https://aclanthology.org/D19-1410/

[5] W. Hamilton, J. Leskovec, and D. Jurafsky, “Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change,” in *Proc. ACL*, 2016. [Online]. Available: https://aclanthology.org/P16-1141/

[6] D. Kobak and P. Berens, “The art of using t-SNE for single-cell transcriptomics,” *Nature Communications*, 2019. doi: 10.1038/s41467-019-13056-x. [Online]. Available: https://doi.org/10.1038/s41467-019-13056-x

[7] J. Grimmer and B. M. Stewart, “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts,” *Political Analysis*, 2013. doi: 10.1093/pan/mps028. [Online]. Available: https://doi.org/10.1093/pan/mps028

[8] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, 1987. doi: 10.1016/0377-0427(87)90125-7. [Online]. Available: https://doi.org/10.1016/0377-0427(87)90125-7