

Text Mining TA #3

Margherita Philipp

Generating text-based features

- Want a time-varying hotel characteristic derived from text
- Should plausibly relate to price levels
- Should also be
 - Transparently derived
 - Quick to generate
 - Stable over time
 - Easy to defend in a DiD setting

Try: feature built with dictionary-based method on luxury language in hotel descriptions

Not yet: vectors

Steps (1/4)

- Minimally clean the text, e.g.
 - Lowercasing
 - Remove punctuation
 - Replace digits, underscores (i.e. not word characters) and whitespace with regular space
 - Strip whitespace at start and end

Any cleaning caveats you can think of?

- U.S. vs us
 - COVID-19 vs covid 19
 - C++ vs c
- could use a tokenizer to deal with punctuation intelligently

Steps (2/4)

- Think of some luxurious words to describe hotels
- If you're feeling extra:
 - Check whether they are in the text and potentially indicative via TFIDF

Steps (3/4)

```
luxury_pattern = re.compile(  
    r"\b(" + "|".join(map(re.escape, luxury_terms)) + r")\b"  
)
```

- Compile regex pattern
 - map(re.escape, luxury_terms) - escapes any special regex characters (+, ., (, etc.)
 - "|".join(...) - places “OR” between joined spaced terms
 - "(" + ... + ")" - wrapt for capturing group (*any* one of the words)
 - r"\b" ... r"\b" - match whole words
 - re.compile(...) - compile once
 - luxury_pattern.search(text)
 - luxury_pattern.findall(text)
- Count matches (can also show words matched)
 - Loop vs pandas vectorized regex approach
- Normalise
 - (matches / text length)

Race time!

Steps (4/4)

- Make the feature :)
- Maybe something binary?
 - How could you decide on where to set the threshold?

You can think of dictionary-based methods as “putting full weight” on a few TFIDF columns

Complete section 1) Text features

- Add your own luxury words
- How do your speed results compare to your neighbour's?

Extra

- Which luxury words get the most matches?
- Where and what are the most luxurious hotels? Are they also the most expensive?

Mini preview

```
from sentence_transformers import SentenceTransformer  
  
model = SentenceTransformer("all-MiniLM-L6-v2")  
df[\"embedding\"] = df[\"text\"].apply(model.encode)
```

- Transformer places each description into vector (e.g. 384-dim)
- Model is static: weights are frozen
 - Same sentence gives same vector
 - embedding dimensions do not have intrinsic meaning
- Never use raw embeddings directly, usually
 - PCA
 - cosine similarity
 - clustering
- So relational meaning can vary
 - PCA component 1 in sample A \neq PCA component 1 in sample B
 - “Luxury-like” direction emerges only relative to other texts
 - Can keep those stable, but then may miss new vocab

Suitability as feature in DiD and beyond?

	Dictionary	Embeddings
Pros		
Cons		

In the name of explainability

	Dictionary	Embeddings
Pros	<p>Fully interpretable</p> <p>Easy to justify in a causal design</p> <p>Easy to debug (you can show the matched words)</p> <p>Somewhat more stable over time</p>	<p>Capture nuance and synonymy</p> <p>Strong predictive performance</p>
Cons	<p>Miss synonyms you didn't think of</p> <p>Crude semantic understanding</p>	<p>Hard to explain what changed</p> <p>Latent dimensions may drift over time</p> <p>Require PCA to reduce dimensions</p>

Diff in diff reminder: when do we use it?

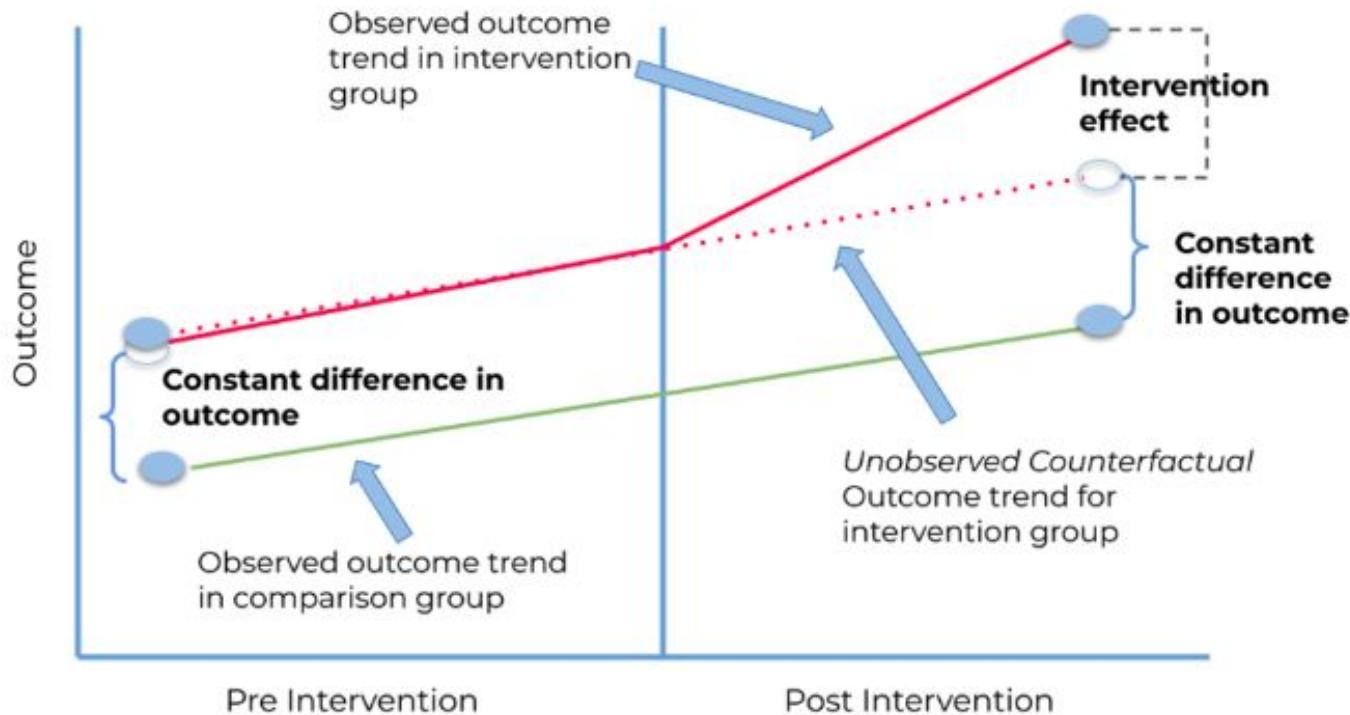
- Randomization is impossible
- Treatment happens at a known time
- You have panel or repeated cross-section data

Diff in diff reminder: what do we need?

- A treatment group and a control group
 - Treatment group: exposed to some intervention/policy/event
 - Control group: not exposed
 - Crucially: treatment is not chosen by the outcome (or you argue it's plausibly exogenous)
- Time variation: before and after treatment
 - At least two time periods:
 - Pre-treatment
 - Post-treatment
 - More periods are better (for testing assumptions and dynamics)
- Parallel trends assumption (the key one)
 - In the absence of treatment, the average outcome of treated and control groups would have evolved in parallel over time.
 - This is not testable directly, but you can:
 - Inspect pre-treatment trends
 - Use event-study plots
 - Argue institutionally / theoretically

Diff in diff reminder

Need to account
for general trend
by finding a
suitable
comparison:



Complete section 2) Check Parallel trends

- Compare visuals for each group
- What “story” could you tell from what you see?

Adding the luxury feature to the regression

$$\ln(\text{price}) = \beta \cdot \text{treatment} + FE + \varepsilon$$

- FE options: date, city, hotel

$$\ln(\text{price}) = \beta \cdot \text{treatment} + \gamma \cdot (\text{treatment} \times \text{luxury}) + FE + \varepsilon$$

- FE options: which of the above don't make sense anymore?

Results (all groups combined)

Python - all groups				
VARIABLES	(3) Inprice	(4) Inprice	(5) Inprice	(6) Inprice
treatment	0.229 0.054	0.176 0.062	0.247 0.033	0.229 0.054
Constant				
(none - because residualized variables have mean ~0)				
Observation	20,177	20,177	20,177	20,177
R-squared				
Control				Luxury
Interaction			Luxury	
FE	City + Date	Hotel + Date	City + Date	City + Date
SE	Cluster(city)	Cluster(city)	Cluster(city)	Cluster(city)

Baseline DiD: city FE + date FE $\rightarrow 0.229$

- within-city changes over time

Hotel FE + date FE $\rightarrow 0.176$ (smaller)

- compare the same hotel to itself over time
- Part of the 0.229 was coming from between-hotel composition effects within cities

Heterogeneous DiD: Treatment \times Luxury $\rightarrow 0.247$

- Luxury hotels respond more strongly to the event?

Adding luxury as a control only \rightarrow still 0.229

- Luxury is time-invariant (or mostly so) at the hotel level (Luxury is correlated with price levels, but not with the timing of the event)

Compare the overall results

- Compare visuals for each group
- What “story” could you tell from what you see?