







# **Classic Embeddings: LSA & LDA**

## **Lecture 9: The Foundations of Distributional Semantics**

**PSYC 51.07: Models of Language and Communication - Week 3**

Winter 2026

# Today's Lecture

1.  The Distributional Hypothesis
2.  Vector Space Models
3.  Latent Semantic Analysis (LSA)
4.  Latent Dirichlet Allocation (LDA)
5.  Modern Topic Modeling: BERTopic
6.  Evaluation & Applications

*Goal: Understand how classic methods represent meaning through co-occurrence*

# The Fundamental Question

**How do we represent the meaning of words in a way that computers can understand?**

\pause

## **Traditional Approach:**

- Dictionaries
- Taxonomies (WordNet)
- Manual feature engineering
- Knowledge bases

Labor-intensive, hard to scale

## **Distributional Approach:**

- Learn from data

# The Distributional Hypothesis

*"You shall know a word by the company it keeps"*

--- J.R. Firth (1957)

\pause

## Key Idea:

Words that appear in similar contexts tend to have similar meanings.

## Example

- "The **cat** sat on the mat"
- "The **dog** sat on the mat"
- "The **kitten** sat on the mat"

→ *cat, dog, kitten* are semantically related

# From Words to Vectors

**Goal:** Represent each word as a point in high-dimensional space

Concrete Example

Consider a tiny corpus with 3 documents:

- Doc 1: "The cat sat on the mat"
- Doc 2: "The dog ran in the park"
- Doc 3: "The cat and dog played"

**Word-context co-occurrence:**

Word	appears with "the"	appears with "sat"	appears with "ran"
cat	2	1	0
dog	2	0	1

# Building a Term-Document Matrix

**Step 1:** Count how often each word appears in each document

cat	0	4	1	0
computer	0	0	0	5
code	0	0	0	3
animal	2	3	2	0

## Issues:

- Very sparse (mostly zeros)
- High dimensional (vocab size  $\times$  doc count)
- Doesn't capture semantic relationships

# TF-IDF Weighting

## Term Frequency - Inverse Document Frequency

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

### Term Frequency (TF):

$$\text{TF}(t, d) = \frac{\text{count}(t, d)}{\sum_{t'} \text{count}(t', d)}$$

How often does term  $t$  appear in document  $d$ ?

*Rewards frequent terms in document*

### Inverse Document Frequency (IDF):

$$\text{IDF}(t) = \log \frac{N}{|\{d : t \in d\}|}$$

# TF-IDF: Worked Example 12 34

**Corpus:** 3 documents, 1000 total documents in collection

Document	Text
Doc 1	"machine learning is great"
Doc 2	"deep learning models"
Doc 3	"machine translation works"

**Calculate TF-IDF for "learning" in Doc 1:**

1Step 1:  $TF("learning", Doc1) = 1/4 = 0.25$  (1 occurrence, 4 words)

2

3Step 2:  $IDF("learning") = \log(1000/500) = \log(2) = 0.301$

4 (appears in 500 of 1000 docs)

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000



# Latent Semantic Analysis (LSA)

The OG of semantic embeddings (1990)

**Algorithm:**

1. Build term-document matrix  $X$
2. Apply TF-IDF weighting
3. Perform SVD:  $X = U\Sigma V^T$
4. Keep top  $k$  dimensions
5. Use  $U_k$  as word embeddings

**Key Innovation:**

- Discovers latent topics
- Solves synonymy problem

# LSA: Step-by-Step Worked Example

Mini corpus (4 words × 3 documents):

	Doc1	Doc2	Doc3
1			
2cat	2	0	1
3dog	0	3	1
4pet	1	2	0
5animal	1	1	1

Step 1: Apply SVD  $X = U\Sigma V^T$

Step 2: Keep top 2 dimensions (k=2)

```

1from sklearn.decomposition import TruncatedSVD
2import numpy as np
3
4X = np.array([[2,0,1], [0,3,1], [1,2,0], [1,1,1]])

```

# LSA: How It Works

## Example: Discovering latent topics

### Original Dimensions:

- 50,000 terms
- 10,000 documents
- Matrix: 50k × 10k
- Sparse and noisy

### After LSA:

- 100-300 latent dimensions
- Dense representation
- Captures semantic patterns

# LSA Limitations

## Computational Issues:

- SVD is expensive:  $O(\min(nm^2, n^2m))$
- Hard to update with new documents
- Memory intensive for large corpora

## Modeling Issues:

- Linear relationships only
- Bag-of-words (loses word order)
- No polysemy handling
- Negative values (hard to interpret)

## Example Problem: Polysemy

# Latent Dirichlet Allocation (LDA)

**A probabilistic approach to topic modeling**

*What if we model documents as probabilistic mixtures of topics?*

## **Key Assumptions:**

1. Each document is a mixture of topics
2. Each topic is a distribution over words
3. The mixture proportions vary per document

## **Advantages over LSA:**

- Probabilistic interpretation
- Non-negative weights

# LDA: The Generative Story

How LDA imagines documents are created:

**Generative Process:**

1. Choose number of topics  $K$
2. For each topic  $k$ :
  - Draw word distribution  $\phi_k \sim \text{Dir}(\beta)$
3. For each document  $d$ :
  - Draw topic distribution  $\theta_d \sim \text{Dir}(\alpha)$
  - For each word position  $n$ :
    - Choose topic  $z_{dn} \sim \text{Mult}(\theta_d)$
    - Choose word  $w_{dn} \sim \text{Mult}(\phi_{z_{dn}})$

# LDA: Concrete Generative Example

Imagine generating a document about "tech pets":

```

1Step 1: Pick topic mixture for this document
2       $\theta_{\text{doc}} = [0.6 \text{ Tech}, 0.3 \text{ Animals}, 0.1 \text{ Sports}]$ 
3
4Step 2: For each word, sample a topic, then sample a word:
5
6Word 1: Sample topic  $\rightarrow$  Tech (60% chance)
7      Sample word from Tech  $\rightarrow$  "software"
8
9Word 2: Sample topic  $\rightarrow$  Animals (30% chance)
10     Sample word from Animals  $\rightarrow$  "cat"
11
12Word 3: Sample topic  $\rightarrow$  Tech (60% chance)
13     Sample word from Tech  $\rightarrow$  "computer"
14
15Word 4: Sample topic  $\rightarrow$  Animals (30% chance)
16     Sample word from Animals  $\rightarrow$  "pet"
    
```

# LDA in Practice

```
1 from sklearn.decomposition import LatentDirichletAllocation
2 from sklearn.feature_extraction.text import CountVectorizer
3
4 # Prepare documents
5 documents = [
6     "The cat sat on the mat",
7     "The dog ran in the park",
8     "Machine learning is amazing",
9     # ... more documents
10 ]
11
12 # Create bag-of-words representation
13 vectorizer = CountVectorizer(max_features=1000, stop_words='english')
14 doc_term_matrix = vectorizer.fit_transform(documents)
15 vocab = vectorizer.get_feature_names_out()
16
17 # Fit LDA model
18 lda = LatentDirichletAllocation(
```



## LDA in Practice

```
21     max_iter=50
22)
23
24doc_topics = lda.fit_transform(doc_term_matrix)
25
26# Print top words per topic
27for idx, topic in enumerate(lda.components_):
28    top_words = [vocab[i] for i in topic.argsort()[-10:]]
29    print(f"Topic {idx}: {' '.join(top_words)}")
```

...continued

# LDA Example: Topic Discovery

## Example Topics from News Corpus:

Topic 1: Politics (20%)

government, election, president, vote, policy, senate, congress, party

Topic 2: Sports (15%)

game, team, player, score, win, season, coach, championship

Topic 3: Technology (25%)

software, computer, data, algorithm, system, internet, code, AI

Topic 4: Finance (40%)

market, stock, price, investment, economy, trade, bank, profit

**Document representation:** [0.20, 0.15, 0.25, 0.40] ← mainly finance

# Modern Topic Modeling: BERTopic

**Combining neural embeddings with topic models**

**BERTopic Pipeline:**

1. Embed documents with BERT
2. Reduce dimensionality with UMAP
3. Cluster with HDBSCAN
4. Generate topics via c-TF-IDF
5. Fine-tune with MaximalMarginalRelevance

**Advantages:**

- Leverages pre-trained models
- Better semantic understanding
- Automatic topic number detection

# Modern Topic Modeling: BERTopic

```
21# Get topic info
22topic_info = model.get_topic_info()
23
24# Visualize
25fig = model.visualize_topics()
```

...continued

*Reference: Grootendorst (2022) - "BERTopic: Neural topic modeling with a class-based TF-IDF procedure"*

# LSA vs. LDA vs. BERTopic

Interpretability	Medium	High	High
Topic coherence	Medium	High	Very High
Computation	Fast	Medium	Slow
Scalability	Good	Good	Medium
Pre-training	No	No	Yes
Hyperparameters	Few	Many	Many
Topic number	Fixed	Fixed	Automatic

## Recommendations

- **LSA**: Quick exploration, large-scale retrieval
- **LDA**: Interpretable topics, when you know topic count

# Evaluating Topic Models

## Intrinsic Metrics:

- **Perplexity:** Lower is better
  - Measures likelihood
  - Can be misleading!
- **Topic Coherence:** Higher is better
  - Measures semantic similarity
  - Better correlation with human judgment
  - CV, UCI, UMass variants
- **Topic Diversity:**
  - Unique words across topics
  - Avoids redundant topics

# Coherence Evaluation: Code Example

```
1 from gensim.models import LdaModel
2 from gensim.models.coherencemodel import CoherenceModel
3 from gensim.corpora import Dictionary
4
5 # Prepare corpus
6 texts = [doc.split() for doc in documents]
7 dictionary = Dictionary(texts)
8 corpus = [dictionary.doc2bow(text) for text in texts]
9
10 # Train LDA with different topic numbers
11 coherence_scores = []
12 for num_topics in [5, 10, 15, 20, 25]:
13     lda = LdaModel(corpus, num_topics=num_topics,
14                   id2word=dictionary, passes=10)
15
16     # Calculate coherence (C_V is recommended)
17     coherence = CoherenceModel(model=lda, texts=texts,
18                               dictionary=dictionary,
```

## Coherence Evaluation: Code Example

```
21
22# Results: [(5, 0.42), (10, 0.51), (15, 0.48), (20, 0.45), (25, 0.41)]
23# Best: 10 topics with coherence 0.51
```

...continued

**Rule of thumb:** Higher coherence = more interpretable topics



# Applications of Classic Embeddings

## 1. Information Retrieval

- Semantic search
- Document similarity
- Query expansion

## 2. Document Organization

- Clustering
- Topic discovery
- Trend analysis

## 3. Text Mining

- Opinion mining
- Literature review

# Application Example: Semantic Search with LSA

```
1 from sklearn.decomposition import TruncatedSVD
2 from sklearn.metrics.pairwise import cosine_similarity
3
4 # Documents already vectorized with TF-IDF
5 # tfidf_matrix shape: (1000 docs, 5000 words)
6
7 # Apply LSA to reduce dimensions
8 lsa = TruncatedSVD(n_components=100)
9 doc_embeddings = lsa.fit_transform(tfidf_matrix)
10
11 # User query: "machine learning algorithms"
12 query_tfidf = vectorizer.transform(["machine learning algorithms"])
13 query_embedding = lsa.transform(query_tfidf)
14
15 # Find most similar documents
16 similarities = cosine_similarity(query_embedding, doc_embeddings)[0]
17 top_5_docs = similarities.argsort()[-5:][::-1]
18
```

## Application Example: Semantic Search with LSA

```
21     print(f"  Doc {idx}: similarity = {similarities[idx]:.3f}")
```

...continued

**Key insight:** LSA finds documents about "neural networks" and "deep learning" even though those exact words weren't in the query!

# Discussion Question

**Can distributional semantics truly capture meaning?**

**Arguments For:**

- Captures semantic similarity
- Works well in practice
- Aligns with linguistic theory
- Scalable to large corpora
- Unsupervised learning

**Arguments Against:**

- Text-only (no grounding)
- Missing embodied experience

# Practical Tips

## 1. Preprocessing Matters:

- Remove stopwords (but not always!)
- Lemmatization vs. stemming
- Handle punctuation carefully
- Consider bigrams/trigrams

## 2. Choosing Parameters:

- Start with 10-50 topics for LDA
- Use perplexity for model selection
- Validate with coherence scores
- Try multiple random seeds

## 3. Interpreting Results:

# Summary

## What we learned today:

1. **Distributional Hypothesis:** Words in similar contexts have similar meanings

2. **LSA (1990):**

- SVD on term-document matrix
- Linear dimensionality reduction
- Fast but limited interpretability

3. **LDA (2003):**

- Probabilistic topic modeling
- Documents as mixtures of topics
- Highly interpretable

4. **BERTopic (2022)**

# Key References

## Classic Papers:

- Firth, J.R. (1957). "A synopsis of linguistic theory 1930-1955"
- Deerwester et al. (1990). "Indexing by Latent Semantic Analysis"
- Blei et al. (2003). "Latent Dirichlet Allocation"
- Harnad, S. (1990). "The symbol grounding problem"

## Modern Extensions:

- Boleda, G. (2020). "Distributional Semantics and Linguistic Theory"
- Grootendorst, M. (2022). "BERTopic: Neural topic modeling with a class-based TF-IDF procedure"
- Grand et al. (2022). "Semantic projection recovers rich human knowledge of multiple object features"

# Questions?

## Next Lecture:

Word Embeddings: Word2Vec, GloVe, FastText

*From count-based to prediction-based methods!*