-->

Language and AI: Course Introduction

Document Type	Value
Course	Language and AI
Institution	Tilburg University, Cognitive Science & AI
Scope	Chapters 1–7 (Weeks 1–7)
Related Materials	Week 1, Week 2, Week 3, Week 4, Week 5, Week 6, Week 7

Index

Brief Course Summary

This course provides a systematic introduction to Natural Language Processing (NLP) and Text Mining, progressing from foundational text representation techniques to state-of-the-art Large Language Models (LLMs). The seven chapters trace the evolution of computational approaches to language understanding: from symbolic preprocessing and sparse vector representations (Weeks 1–2), through probabilistic classification and n-gram language models (Week 3), to distributed word embeddings and neural network fundamentals (Week 4). The course then advances to sequence labeling with Hidden Markov Models and Conditional Random Fields (Week 5), deep learning architectures including RNNs and Transformers (Week 6), and culminates with Large Language Models, pretraining paradigms, and prompting strategies (Week 7). Throughout, mathematical rigor is balanced with practical application, equipping students with both theoretical foundations and implementable techniques.

Abstract

The Language and AI course offers a comprehensive treatment of Natural Language Processing, spanning seven interconnected chapters that collectively address the transformation of human language into computational representations, the extraction of patterns from textual data, and the generation of coherent text. The curriculum begins with fundamental challenges: tokenizing raw text, constructing vector space models via TF-IDF weighting, and quantifying document similarity through Euclidean distance, Jaccard coefficients, and cosine similarity. Preprocessing techniques—regular expressions, stemming, lemmatization, and subword tokenization via Byte-Pair Encoding—establish the data pipeline. Probabilistic methods are then introduced through n-gram language models and Naive Bayes classification, with smoothing techniques addressing data sparsity. The transition to distributed representations is realized through Word2Vec embeddings and the introduction of neural

network fundamentals including feedforward architectures, backpropagation, and cross-entropy optimization. Sequence labeling formalizes token-level prediction via Hidden Markov Models and the Viterbi algorithm, with Conditional Random Fields enabling discriminative feature engineering. Deep learning for NLP advances through Recurrent Neural Networks (capturing sequential dependencies), LSTMs (mitigating vanishing gradients), encoder-decoder architectures (enabling sequence-to-sequence transduction), and the attention mechanism (addressing representational bottlenecks). The course culminates with Transformer-based Large Language Models, distinguishing causal (GPT) from masked (BERT) pretraining objectives, and examining modern deployment strategies including instruction tuning, RLHF, and prompting paradigms.

Keywords

Natural Language Processing; Text Mining; Tokenization; TF-IDF; Cosine Similarity; Regular Expressions; Minimum Edit Distance; N-gram Language Models; Naive Bayes; Logistic Regression; Word Embeddings; Word2Vec; Neural Networks; Backpropagation; Sequence Labeling; Hidden Markov Models; Viterbi Algorithm; Conditional Random Fields; Recurrent Neural Networks; LSTM; Attention Mechanism; Transformers; BERT; GPT; Large Language Models; Pretraining; Finetuning; Prompting

---

1. Introduction

1.1 Course Philosophy

This course is grounded in the principle that effective application of modern NLP techniques requires understanding their theoretical foundations. While contemporary Large Language Models achieve remarkable performance through end-to-end learning, practitioners benefit from comprehending the historical trajectory and mathematical underpinnings of the field. Each chapter builds upon its predecessors, establishing concepts that reappear—often in more sophisticated forms—in subsequent material.

The course emphasizes three complementary perspectives:

1. **Mathematical Formalism**: Key algorithms are derived rather than merely stated, with attention to probability theory, optimization, and linear algebra.

2. **Historical Context**: The evolution from symbolic to statistical to neural paradigms is traced, illuminating why certain approaches succeeded while others were superseded.
3. **Practical Application**: Theoretical concepts are grounded in concrete NLP tasks—sentiment analysis, named entity recognition, machine translation—demonstrating their real-world utility.

1.2 Learning Trajectory

The seven chapters follow a deliberate progression:

- **Weeks 1–2** establish the data pipeline: transforming raw text into structured representations suitable for machine learning.
- **Week 3** introduces probabilistic reasoning: modeling language as sequences of random variables and classifying documents based on word distributions.
- **Week 4** bridges classical and neural approaches: dense word embeddings capture semantic relationships, while neural network fundamentals prepare for deep learning.
- **Week 5** extends classification to sequences: each token receives a label, with dependencies modeled via probabilistic graphical models.
- **Week 6** introduces deep learning architectures: RNNs model temporal dependencies, while Transformers enable parallelizable attention-based processing.
- **Week 7** synthesizes the preceding material in the context of Large Language Models, addressing pretraining, finetuning, and prompting.

---

2. Chapter Summaries and Key Takeaways

2.1 Chapter 1: Foundations of Text Representation and Similarity

Scope: Week 1 | **Primary Reference**: Chapter 1 (Jurafsky & Martin)

Summary: This chapter establishes the mathematical and conceptual foundations for processing natural language computationally. It addresses the fundamental challenge of converting unstructured text into numerical representations amenable to machine learning algorithms. The Bag-of-Words (BoW) model is introduced as the canonical approach to text vectorization, with three weighting schemes examined: binary representation, term frequency (TF), and Term Frequency–Inverse Document Frequency (TF-IDF). The chapter then formalizes

similarity and distance measures—Euclidean distance, Jaccard coefficient, and cosine similarity—that enable quantitative comparison of documents in vector space.

Key Takeaways:

- Language presents unique computational challenges including lexical ambiguity, syntactic structure, and pragmatic context.
- The **Bag-of-Words model** discards word order, representing documents as unordered collections of terms.
- **TF-IDF weighting** ($w_{t,d} = \ln(\text{tf}(t,d) + 1) \cdot \log_{10}(N/\text{df}_t)$) upweights discriminative terms while downweighting ubiquitous ones.
- **Cosine similarity** measures angular proximity between document vectors, providing length-invariant comparison.
- ℓ_2 **normalization** is essential preprocessing for length-independent document comparison.

2.2 Chapter 2: Regular Expressions, Text Normalization, and Edit Distance

Scope: Week 2 | **Primary Reference**: Chapter 2 (Jurafsky & Martin)

Summary: This chapter addresses the preprocessing pipeline that transforms raw text into structured representations. Regular expressions are introduced as a formal language for pattern specification, enabling precise text search, extraction, and substitution. Tokenization is examined through both rule-based (top-down) and data-driven (bottom-up) approaches, with particular attention to Byte-Pair Encoding (BPE). Text normalization techniques—case folding, lemmatization, and stemming—reduce vocabulary size and improve generalization. The chapter concludes with the minimum edit distance algorithm, a dynamic programming solution for quantifying string similarity.

Key Takeaways:

- **Regular expressions** provide an algebraic notation for characterizing string sets, with operations including concatenation, disjunction, and Kleene closure.
- **Heaps' Law** ($|V| = kN^{\beta}$) characterizes the sublinear relationship between corpus size and vocabulary growth.

- **Byte-Pair Encoding (BPE)** automatically induces subword vocabularies, handling rare words and morphological variation.
- **Lemmatization** maps inflected forms to canonical dictionary entries; **stemming** applies heuristic suffix stripping.
- The **minimum edit distance** (Levenshtein distance) algorithm computes the optimal sequence of insertions, deletions, and substitutions via dynamic programming: $D[i,j] = \min\{D[i-1,j]+1, D[i,j-1]+1, D[i-1,j-1]+\text{cost}\}$.

2.3 Chapter 3: Classification and Language Modeling

Scope: Week 3 | **Primary Reference**: Chapters 3, 4, 5 (Jurafsky & Martin)

Summary: This chapter presents a unified treatment of probabilistic language modeling and supervised text classification. N-gram language models formalize the estimation of word sequence probabilities under the Markov assumption, with smoothing algorithms (Laplace, add-k, interpolation) addressing the zero-frequency problem. Classification is developed through both generative (Naive Bayes) and discriminative (logistic regression) paradigms. Additional algorithms—Decision Trees, Support Vector Machines, and k-Nearest Neighbors—complete the survey. Evaluation methodology including precision, recall, F-measure, and cross-validation enables rigorous model comparison.

Key Takeaways:

- **N-gram language models** estimate $P(w_n|w_{1:n-1})$ via the chain rule, with the Markov assumption truncating context to $n-1$ preceding words.
- **Perplexity** ($PP(W) = P(W)^{-1/N}$) serves as the standard intrinsic evaluation metric, measuring how "surprised" the model is by test data.
- **Naive Bayes** applies Bayes' theorem with the conditional independence assumption: $\hat{c} = \arg\max_c P(c)\prod_i P(w_i|c)$.
- **Logistic regression** directly models $P(y|\mathbf{x})$ via the sigmoid/softmax function, trained by minimizing cross-entropy loss through gradient descent.
- **Smoothing** (Laplace smoothing) prevents zero probabilities for unseen events: $P_{\text{Laplace}}(w_i) = \frac{C(w_i)+1}{N+V}$.

2.4 Chapter 4: Representation Learning — Vector Semantics and Neural Networks

Scope: Week 4 | **Primary Reference**: Chapters 6, 7 (Jurafsky & Martin)

Summary: This chapter addresses the fundamental question of how to represent meaning computationally. Beginning with the distributional hypothesis ("you shall know a word by the company it keeps"), the chapter develops both count-based vector semantics (TF-IDF, PPMI) and prediction-based word embeddings (Word2Vec Skip-Gram with Negative Sampling). The second half introduces neural network fundamentals: computational units with nonlinear activations, feedforward architectures, cross-entropy loss, and backpropagation through computation graphs. The chapter concludes with feedforward neural language models that predict the next word while acquiring useful representations.

Key Takeaways:

- The **distributional hypothesis** states that words appearing in similar contexts have similar meanings.
- **Pointwise Mutual Information (PMI)** quantifies word association strength: $\text{PMI}(w, c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$; PPMI replaces negative values with zero.
- **Word2Vec Skip-Gram** learns dense embeddings by predicting context words from target words, trained via negative sampling.
- **Activation functions** (sigmoid, tanh, ReLU) introduce nonlinearity enabling multi-layer networks to learn complex decision boundaries.
- **Backpropagation** computes gradients by applying the chain rule backwards through the computation graph.

2.5 Chapter 5: Information Extraction — Sequence Labeling and Structured Prediction

Scope: Week 5 | **Primary Reference**: Chapters 17, 20 (Jurafsky & Martin)

Summary: This chapter addresses the transition from document-level classification to sequence labeling, where each token receives a corresponding label. Part-of-speech tagging and Named Entity Recognition serve as canonical examples, with BIO tagging schemes enabling span detection. Hidden Markov Models are developed as generative sequence models with transition and emission probabilities estimated via Maximum Likelihood. The Viterbi algorithm

provides efficient dynamic programming decoding. Conditional Random Fields are introduced as discriminative alternatives enabling rich feature engineering. The chapter extends to relation extraction, event detection, and temporal analysis.

Key Takeaways:

- **Sequence labeling** assigns labels to each token in a sequence, modeling dependencies between adjacent labels.
- **BIO tagging** (Begin, Inside, Outside) enables recognition of multi-token spans for named entities.
- **Hidden Markov Models** decompose the joint probability: $P(W, T) = \prod_i P(t_i|t_{i-1}) \cdot P(w_i|t_i)$.
- The **Viterbi algorithm** finds the most probable state sequence in $O(TN^2)$ time via dynamic programming: $v_t(j) = \max_i [v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)]$.
- **Conditional Random Fields (CRFs)** directly model $P(Y|X)$, permitting arbitrary overlapping features without independence assumptions.

2.6 Chapter 6: Deep Learning for NLP — Recurrent Networks and Transformers

Scope: Week 6 | **Primary Reference**: Chapter 8 (Jurafsky & Martin)

Summary: This chapter develops deep learning architectures for sequential language processing. Recurrent Neural Networks introduce temporal structure through recurrent connections, enabling language modeling without fixed context windows. Long Short-Term Memory (LSTM) networks address the vanishing gradient problem through gating mechanisms. The encoder-decoder architecture extends RNNs to sequence-to-sequence tasks. The attention mechanism computes dynamic context vectors as weighted sums of encoder states. Finally, Transformers replace recurrence entirely with self-attention, using Query-Key-Value projections and scaled dot-product attention for parallelizable sequence modeling.

Key Takeaways:

- **RNNs** maintain hidden states that carry information across time steps: $h_t = g(Uh_{t-1} + Wx_t)$.

- The **vanishing gradient problem** limits RNNs' ability to capture long-range dependencies; **LSTMs** mitigate this via forget, input, and output gates.
- **Encoder-decoder** architectures map input sequences to fixed-length context vectors consumed by autoregressive decoders.
- The **attention mechanism** computes context vectors as weighted sums of encoder states: $c_i = \sum_j \alpha_{ij} h_j^e$, where weights reflect relevance.
- **Transformers** replace recurrence with **self-attention**: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, enabling parallel computation.

2.7 Chapter 7: Large Language Models

Scope: Week 7 | **Primary Reference**: Chapters 4, 10, 11, 12 (Jurafsky & Martin)

Summary: This chapter provides a comprehensive treatment of Large Language Models (LLMs), integrating foundational text classification with modern transformer-based architectures. The exposition revisits Naive Bayes classification before transitioning to transformer-based LLMs. Both causal (GPT-style) and masked (BERT-style) language models are examined, with emphasis on their distinct pretraining objectives. Contextual embeddings are characterized as dynamic representations enabling word sense disambiguation. The chapter addresses scaling laws, KV caching, and parameter-efficient finetuning (LoRA). Model alignment via instruction tuning and RLHF is presented, alongside prompting paradigms including zero-shot, few-shot, and chain-of-thought approaches. Ethical considerations including hallucination, toxicity, and bias conclude the treatment.

Key Takeaways:

- **Pretraining** acquires language knowledge through self-supervised learning on massive text corpora; **finetuning** adapts models to specific downstream tasks.
- **Causal language models** (GPT family) predict the next token autoregressively; **masked language models** (BERT family) reconstruct masked tokens using bidirectional context.
- **Contextual embeddings** produce position-dependent vector representations where the same word receives different embeddings based on surrounding context.
- **Instruction tuning** and **RLHF** (Reinforcement Learning from Human Feedback) align model outputs with human preferences and intended behaviors.

- **Prompting** enables task performance without parameter updates: **zero-shot** (task description only), **few-shot** (task description plus examples), and **chain-of-thought** (explicit reasoning steps).

---

3. Thematic Connections Across Chapters

3.1 Representation: From Sparse to Dense to Contextual

A central thread connecting all chapters is the evolving approach to representing linguistic units:

Stage	Representation	Chapters	Characteristics
Sparse Symbolic	TF-IDF vectors	1–2	High-dimensional, interpretable, context-independent
Dense Static	Word2Vec embeddings	4	Low-dimensional, learned, context-independent
Dense Contextual	BERT/GPT representations	6–7	Position-dependent, polysemy-aware, pretrained

This progression reflects the field's recognition that word meaning is not fixed but emerges from context—a principle formalized in the distributional hypothesis (Chapter 4) and realized most fully in contextual embeddings (Chapter 7).

3.2 Learning Paradigms: From Counting to Prediction

The course traces an evolution in how models acquire knowledge about language:

- **Counting** (Chapters 1–3): Collect co-occurrence statistics and normalize appropriately (TF-IDF, n-gram probabilities).
- **Prediction** (Chapters 4, 6): Learn representations by predicting words from context (Skip-Gram, RNN language models).
- **Pretraining + Finetuning** (Chapter 7): Self-supervised learning on massive corpora followed by task-specific adaptation.

3.3 Architecture Evolution: From Bag-of-Words to Transformers

The treatment of sequence structure evolves progressively:

Model	Word Order	Context	Chapter
Bag-of-Words	Ignored	None	1, 3
N-grams	Local (n tokens)	Fixed window	3
HMMs/CRFs	Sequential	Previous label	5
RNNs/LSTMs	Sequential	All previous tokens (theoretically)	6
Transformers	Encoded via positional embeddings	All tokens (bidirectionally)	6–7

---

4. Conclusion

The Language and AI course provides a structured journey through the foundations and frontiers of Natural Language Processing. Beginning with the challenge of representing text numerically, the course develops increasingly sophisticated techniques for capturing linguistic meaning. The progression from sparse vector spaces to contextual embeddings reflects the field's deepening understanding of how meaning emerges from context. Similarly, the evolution from hand-crafted features to end-to-end learning demonstrates the power of data-driven approaches when coupled with appropriate inductive biases.

Students completing this course will possess both the theoretical foundations to understand current NLP research and the practical skills to implement effective text processing systems. The emphasis on mathematical derivation ensures that techniques are understood rather than merely applied, while the historical perspective illuminates why certain approaches have proven more successful than others.

The course ultimately demonstrates that language—long considered a uniquely human capability—can be modeled computationally with remarkable fidelity. Large Language Models represent not the end of this journey but rather a milestone, with ongoing research addressing their limitations including hallucination, reasoning, and alignment with human values.

---

5. References

- Jurafsky, D. & Martin, J.H. (2024). *Speech and Language Processing* (3rd ed., draft). Stanford University.
- Vaswani, A., et al. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*.

- Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*.
- Mikolov, T., et al. (2013). Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.
- Brown, T., et al. (2020). Language Models are Few-Shot Learners. *NeurIPS*.
- Ouyang, L., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. *NeurIPS*.