UCLA COMPUTER SCIENCE FIELD NOTES

CS 162

Natural Language Processing

Text representation, language models, sequence learning, transformers, NLP tasks, and evaluation

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	**Language tasks, corpora, and evaluation design**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
2	**Tokenization, morphology, and subwords**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
3	**N-gram language models and smoothing**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
4	**Vector semantics and representation learning**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
5	**Sequence models and structured prediction**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
6	**Attention and transformers**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
7	**Core NLP systems and transfer**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
8	**Bias, robustness, privacy, and deployment**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
9	**Integration workshop**	Connect at least three chapters in one end-to-end problem and explain where their contracts meet.
10	**Synthesis and project review**	Audit assumptions and explain a complete solution from interface to failure handling.

Scope anchor: https://catalog.registrar.ucla.edu/course/2025/COMSCI162. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

CHAPTER 1 Language tasks, corpora, and evaluation design

The idea

NLP maps language to representations and decisions for classification, tagging, parsing, translation, retrieval, question answering, summarization, and generation. Corpus construction, annotation, splits, baselines, and metrics define success.

Rebuild the chapter from first principles

The claim I need to own is this: NLP maps language to representations and decisions for classification, tagging, parsing, translation, retrieval, question answering, summarization, and generation. Corpus construction, annotation, splits, baselines, and metrics define success. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of language tasks, corpora, and evaluation design. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach language tasks, corpora, and evaluation design on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to tokenization, morphology, and subwords. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** NLP maps language to representations and decisions for classification, tagging, parsing, translation, retrieval, question answering, summarization, and generation.
- **Mechanism.** Corpus construction, annotation, splits, baselines, and metrics define success.
- **Boundary.** The useful test is whether I can apply language tasks, corpora, and evaluation design to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when language tasks, corpora, and evaluation design is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I define the unit of prediction and cost of a mistake before choosing a model.

The easy version

NLP maps language to representations and decisions for classification, tagging, parsing, translation, retrieval, question answering, summarization, and generation.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why can a random sentence split leak information? The conclusion is Near-duplicate text, the same author, document, conversation, or future information can enter both train and test. Split by the deployment boundary such as user,

document, site, or time.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why can a random sentence split leak information?
Solution. Near-duplicate text, the same author, document, conversation, or future information can enter both train and test. Split by the deployment boundary such as user, document, site, or time.

Practice 2. Give a short oral explanation of language tasks, corpora, and evaluation design that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct language tasks, corpora, and evaluation design from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a language tasks, corpora, and evaluation design problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Tokenization, morphology, and subwords

The idea

Text preprocessing handles normalization, sentence boundaries, words, morphology, rare forms, scripts, and Unicode. Subword methods balance open vocabulary with sequence length, but learned boundaries can differ across languages and groups.

Rebuild the chapter from first principles

The claim I need to own is this: Text preprocessing handles normalization, sentence boundaries, words, morphology, rare forms, scripts, and Unicode. Subword methods balance open vocabulary with sequence length, but learned boundaries can differ across languages and groups. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of tokenization, morphology, and subwords. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach tokenization, morphology, and subwords on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from language tasks, corpora, and evaluation design to n-gram language models and smoothing. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Text preprocessing handles normalization, sentence boundaries, words, morphology, rare forms, scripts, and Unicode.
- **Mechanism.** Subword methods balance open vocabulary with sequence length, but learned boundaries can differ across languages and groups.
- **Boundary.** The useful test is whether I can apply tokenization, morphology, and subwords to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when tokenization, morphology, and subwords is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: a token is a modeling decision, not a natural atom of language.

The easy version

Text preprocessing handles normalization, sentence boundaries, words, morphology, rare forms, scripts, and Unicode.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why do subwords help with unknown words? The conclusion is They compose rare or new words from frequent pieces, allowing shared parameters and a bounded vocabulary. The cost is longer sequences and sometimes awkward segmentation.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why do subwords help with unknown words?
Solution. They compose rare or new words from frequent pieces, allowing shared parameters and a bounded vocabulary. The cost is longer sequences and sometimes awkward segmentation.

Practice 2. Give a short oral explanation of tokenization, morphology, and subwords that includes

its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct tokenization, morphology, and subwords from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a tokenization, morphology, and subwords problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 N-gram language models and smoothing

The idea

A language model assigns probability to token sequences. N-grams approximate history with finite context; maximum-likelihood counts need smoothing and backoff so unseen events receive mass. Perplexity measures fit under fixed tokenization.

Rebuild the chapter from first principles

The claim I need to own is this: A language model assigns probability to token sequences. N-grams approximate history with finite context; maximum-likelihood counts need smoothing and backoff so unseen events receive mass. Perplexity measures fit under fixed tokenization. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of n-gram language models and smoothing. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.

3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach n-gram language models and smoothing on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from tokenization, morphology, and subwords to vector semantics and representation learning. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** A language model assigns probability to token sequences.
- **Mechanism.** N-grams approximate history with finite context; maximum-likelihood counts need smoothing and backoff so unseen events receive mass.
- **Boundary.** Perplexity measures fit under fixed tokenization.

- **Decision test.** I should be able to say when n-gram language models and smoothing is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I check the counting denominator and vocabulary before trusting a probability or perplexity comparison.

The easy version

A language model assigns probability to token sequences.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why is add-one smoothing often too aggressive? The conclusion is It reserves substantial mass for every unseen n-gram, especially with a large vocabulary, and distorts frequent estimates. Discounting and interpolation allocate unseen mass more carefully.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why is add-one smoothing often too aggressive?
Solution. It reserves substantial mass for every unseen n-gram, especially with a large vocabulary, and distorts frequent estimates. Discounting and interpolation allocate unseen mass more carefully.

Practice 2. Give a short oral explanation of n-gram language models and smoothing that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct n-gram language models and smoothing from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a n-gram language models and smoothing problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 Vector semantics and representation learning

The idea

Distributional semantics represents meaning through usage context. Count vectors, TF-IDF, matrix factorization, static embeddings, and contextual embeddings trade interpretability, sparsity, polysemy handling, computation, and data dependence.

Rebuild the chapter from first principles

The claim I need to own is this: Distributional semantics represents meaning through usage context. Count vectors, TF-IDF, matrix factorization, static embeddings, and contextual embeddings trade interpretability, sparsity, polysemy handling, computation, and data dependence. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of vector semantics and representation learning. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.

- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach vector semantics and representation learning on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from n-gram language models and smoothing to sequence models and structured prediction. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Distributional semantics represents meaning through usage context.
- **Mechanism.** Count vectors, TF-IDF, matrix factorization, static embeddings, and contextual embeddings trade interpretability, sparsity, polysemy handling, computation, and data dependence.
- **Boundary.** The useful test is whether I can apply vector semantics and representation learning to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when vector semantics and representation learning is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: a nearby vector means similar use under training data, not factual or causal equivalence.

The easy version

Distributional semantics represents meaning through usage context.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why can one static word vector mishandle polysemy? The conclusion is All senses share one representation, so finance and river uses of 'bank' are averaged. Contextual models condition the representation on surrounding tokens.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why can one static word vector mishandle polysemy?
Solution. All senses share one representation, so finance and river uses of 'bank' are averaged. Contextual models condition the representation on surrounding tokens.

Practice 2. Give a short oral explanation of vector semantics and representation learning that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct vector semantics and representation learning from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a vector semantics and representation learning problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Sequence models and structured prediction

The idea

Hidden Markov models, conditional random fields, recurrent networks, LSTMs, and encoder-decoder models represent sequence dependence. Structured decoding enforces relationships among labels that independent token predictions can violate.

Rebuild the chapter from first principles

The claim I need to own is this: Hidden Markov models, conditional random fields, recurrent networks, LSTMs, and encoder-decoder models represent sequence dependence. Structured decoding enforces relationships among labels that independent token predictions can violate. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of sequence models and structured prediction. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach sequence models and structured prediction on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from vector semantics and representation learning to attention and transformers. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Hidden Markov models, conditional random fields, recurrent networks, LSTMs, and encoder-decoder models represent sequence dependence.
- **Mechanism.** Structured decoding enforces relationships among labels that independent token predictions can violate.
- **Boundary.** The useful test is whether I can apply sequence models and structured prediction to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when sequence models and structured prediction is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I separate the score for one label from the score for a legal complete sequence.

The easy version

Hidden Markov models, conditional random fields, recurrent networks, LSTMs, and encoder-decoder models represent sequence dependence.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why use Viterbi decoding for a sequence tagger? The conclusion is It finds the highest-scoring legal label path while including transition

scores between adjacent labels, rather than choosing each token independently.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why use Viterbi decoding for a sequence tagger?
Solution. It finds the highest-scoring legal label path while including transition scores between adjacent labels, rather than choosing each token independently.

Practice 2. Give a short oral explanation of sequence models and structured prediction that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct sequence models and structured prediction from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a sequence models and structured prediction problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Attention and transformers

The idea

Self-attention builds each token representation from weighted interactions with other tokens. Queries, keys, values, heads, position information, residual connections, normalization, masking, and feed-forward layers form a transformer block.

Rebuild the chapter from first principles

The claim I need to own is this: Self-attention builds each token representation from weighted interactions with other tokens. Queries, keys, values, heads, position information, residual connections, normalization, masking, and feed-forward layers form a transformer block. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of attention and transformers. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach attention and transformers on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from sequence models and structured prediction to core nlp systems and transfer. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Self-attention builds each token representation from weighted interactions with other tokens.
- **Mechanism.** Queries, keys, values, heads, position information, residual connections, normalization, masking, and feed-forward layers form a transformer block.
- **Boundary.** The useful test is whether I can apply attention and transformers to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when attention and transformers is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I read attention as learned information routing, not automatically as a faithful explanation.

The easy version

Self-attention builds each token representation from weighted interactions with other tokens.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why does causal attention use a mask? The conclusion is At position t, the model must not read future target tokens during autoregressive training. The mask removes those scores, preserving the next-token prediction contract.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why does causal attention use a mask?
Solution. At position t, the model must not read future target tokens during autoregressive training. The mask removes those scores, preserving the next-token prediction contract.

Practice 2. Give a short oral explanation of attention and transformers that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison

according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct attention and transformers from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a attention and transformers problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Core NLP systems and transfer

The idea

Modern NLP adapts pretrained representations to classification, extraction, retrieval, translation, question answering, summarization, and generation through prompting, fine-tuning, retrieval, or task-specific heads.

Rebuild the chapter from first principles

The claim I need to own is this: Modern NLP adapts pretrained representations to classification, extraction, retrieval, translation, question answering, summarization, and generation through prompting, fine-tuning, retrieval, or task-specific heads. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of core nlp systems and transfer. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.

4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach core nlp systems and transfer on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from attention and transformers to bias, robustness, privacy, and deployment. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Modern NLP adapts pretrained representations to classification, extraction, retrieval, translation, question answering, summarization, and generation through prompting, fine-tuning, retrieval, or task-specific heads.
- **Mechanism.** The useful test is whether I can apply core nlp systems and transfer to a new case and explain the assumption that makes the answer valid.
- **Boundary.** The useful test is whether I can apply core nlp systems and transfer to a new case and

explain the assumption that makes the answer valid.

- **Decision test.** I should be able to say when core nlp systems and transfer is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I keep a simple baseline because a large model can hide leakage behind impressive average performance.

The easy version

Modern NLP adapts pretrained representations to classification, extraction, retrieval, translation, question answering, summarization, and generation through prompting, fine-tuning, retrieval, or task-specific heads.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: How should a summarizer be evaluated beyond overlap score? The conclusion is Check factual consistency, coverage, relevance, redundancy, readability, source support, subgroup and domain behavior, human preference, and task-specific harms.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. How should a summarizer be evaluated beyond overlap score?
Solution. Check factual consistency, coverage, relevance, redundancy, readability, source support, subgroup and domain behavior, human preference, and task-specific harms.

Practice 2. Give a short oral explanation of core nlp systems and transfer that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct core nlp systems and transfer from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a core nlp systems and

transfer problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Bias, robustness, privacy, and deployment

The idea

NLP systems reproduce properties of corpora and annotation. Dialect, language coverage, toxicity, hallucination, prompt sensitivity, privacy leakage, adversarial input, latency, and monitoring affect real quality.

Rebuild the chapter from first principles

The claim I need to own is this: NLP systems reproduce properties of corpora and annotation. Dialect, language coverage, toxicity, hallucination, prompt sensitivity, privacy leakage, adversarial input, latency, and monitoring affect real quality. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of bias, robustness, privacy, and deployment. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.

- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach bias, robustness, privacy, and deployment on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from core nlp systems and transfer to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** NLP systems reproduce properties of corpora and annotation.
- **Mechanism.** Dialect, language coverage, toxicity, hallucination, prompt sensitivity, privacy leakage, adversarial input, latency, and monitoring affect real quality.
- **Boundary.** The useful test is whether I can apply bias, robustness, privacy, and deployment to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when bias, robustness, privacy, and deployment is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

Fluent language lowers my guard, so I demand evidence precisely when the output sounds most confident.

The easy version

NLP systems reproduce properties of corpora and annotation.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: A chatbot performs worse for one dialect. What is the next step? The conclusion is Measure error types with community-informed evaluation, audit data and annotation coverage, improve objectives, test alternatives, add escalation and monitoring, and avoid treating the dialect as defective language.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. A chatbot performs worse for one dialect. What is the next step?
Solution. Measure error types with community-informed evaluation, audit data and annotation coverage, improve objectives, test alternatives, add escalation and monitoring, and avoid treating the dialect as defective language.

Practice 2. Give a short oral explanation of bias, robustness, privacy, and deployment that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct bias, robustness, privacy, and deployment from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a bias, robustness, privacy, and deployment problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Final study check

I should be able to close this packet and reconstruct the core models, not merely recognize their names. My final check is to explain one complete problem aloud: what the inputs mean, which invariant or contract controls the work, why the method is legal, what it costs, how it can fail, and what evidence would convince another engineer.