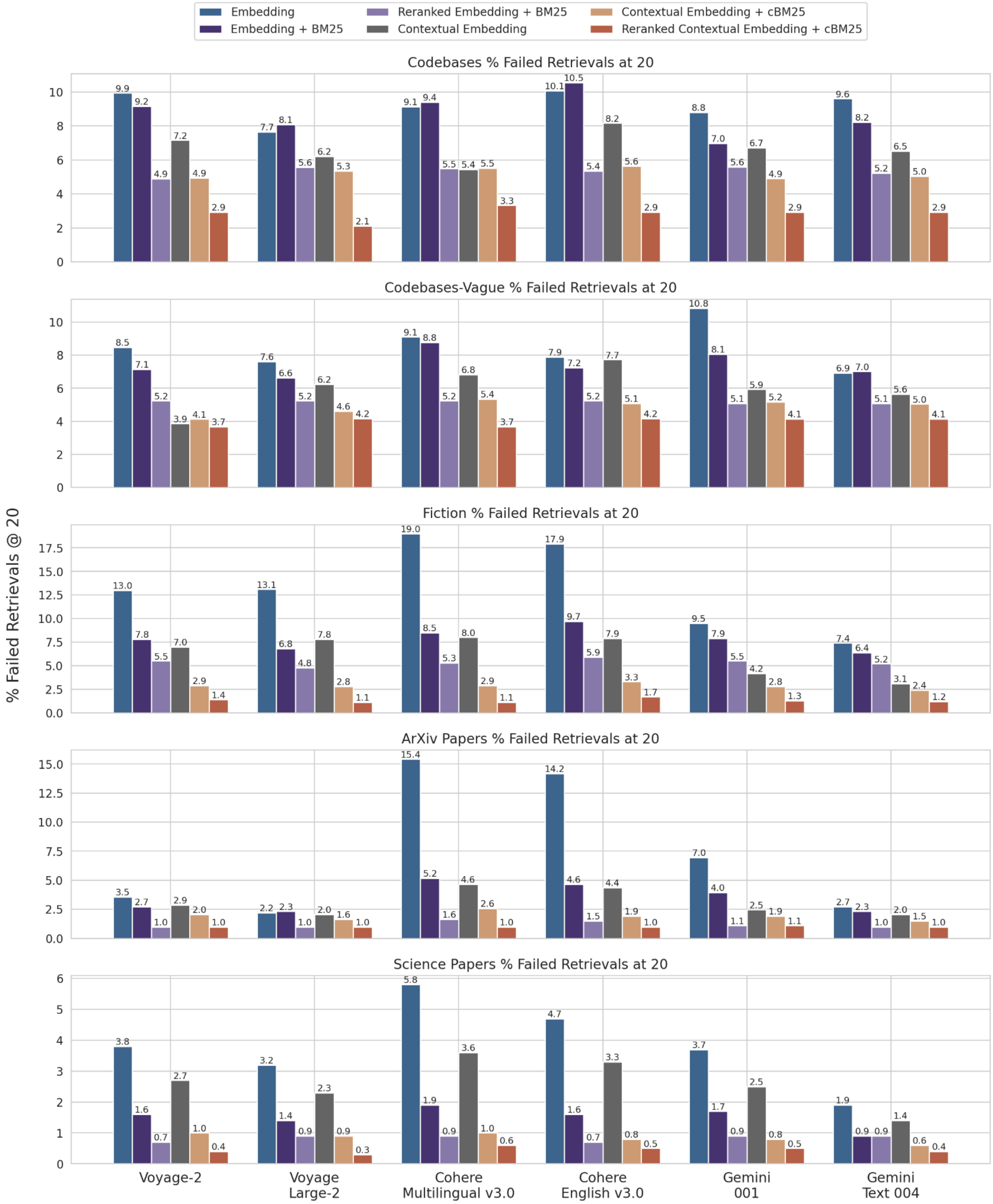
Contextual Retrieval Appendix II

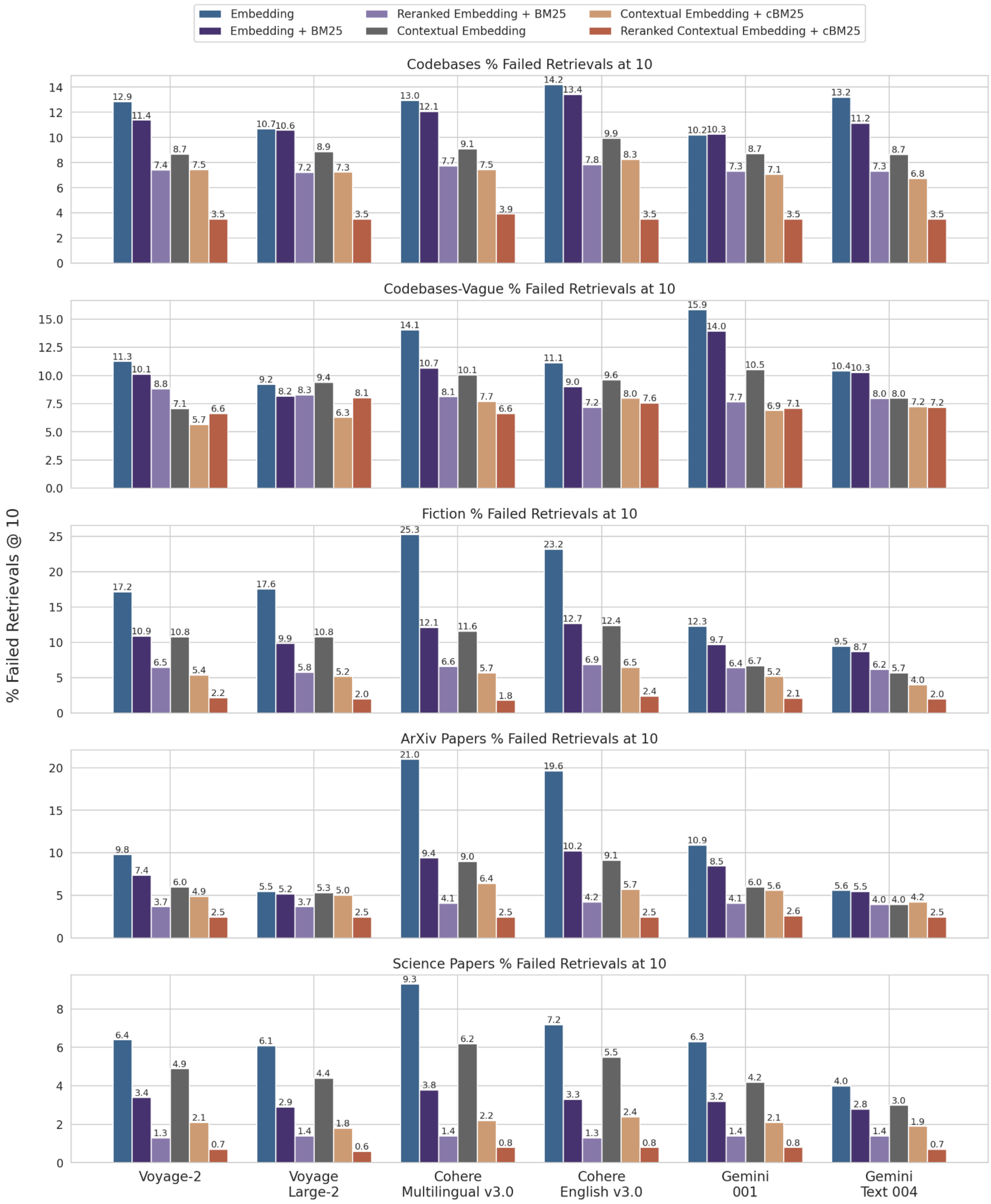
Full breakdown of experiment results

Here is a breakdown of results across datasets, embedding providers, choice of top-k, use of BM25 in addition to embeddings, use of contextual retrieval, and use of reranking.

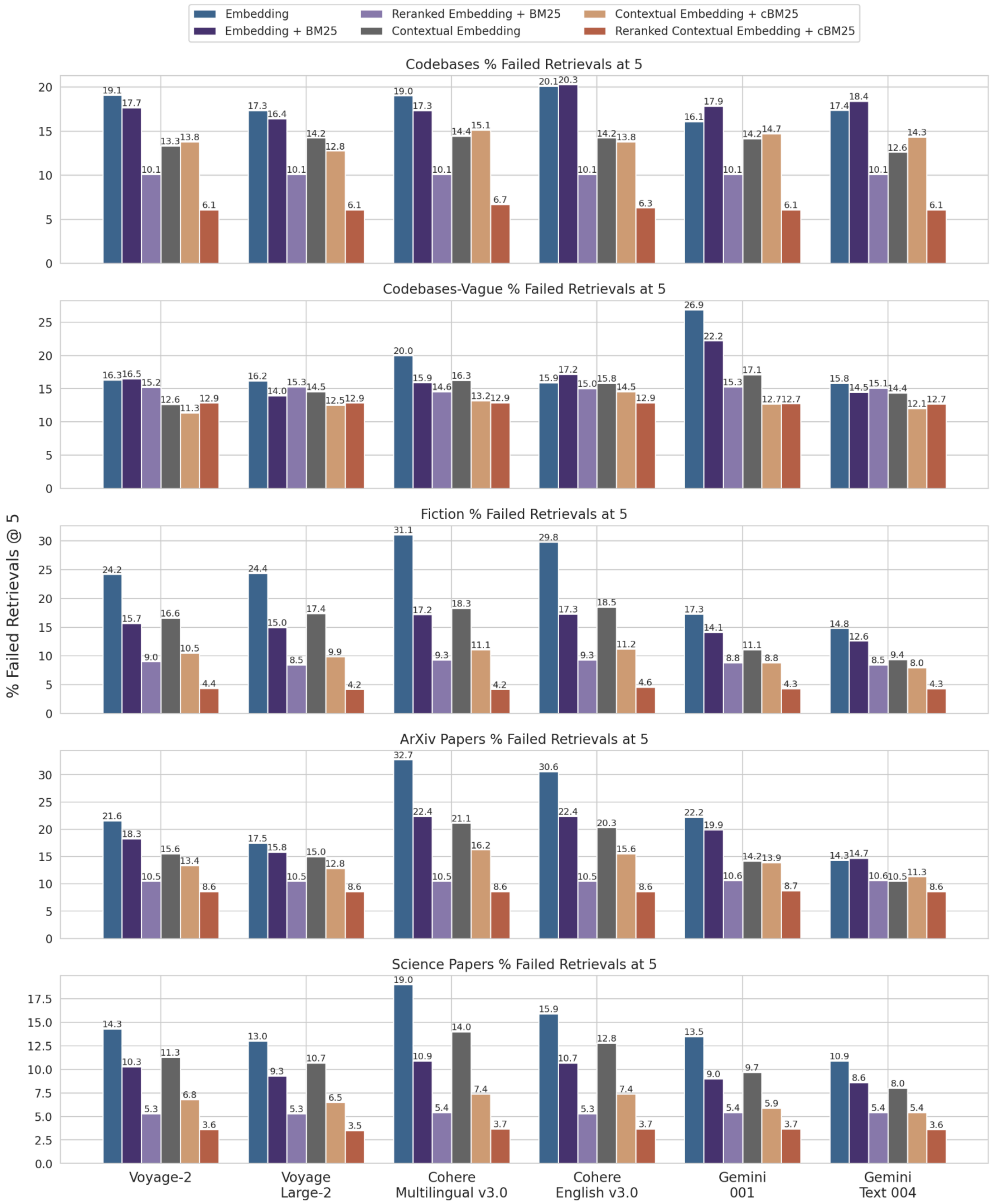
1 minus recall @ 20 results



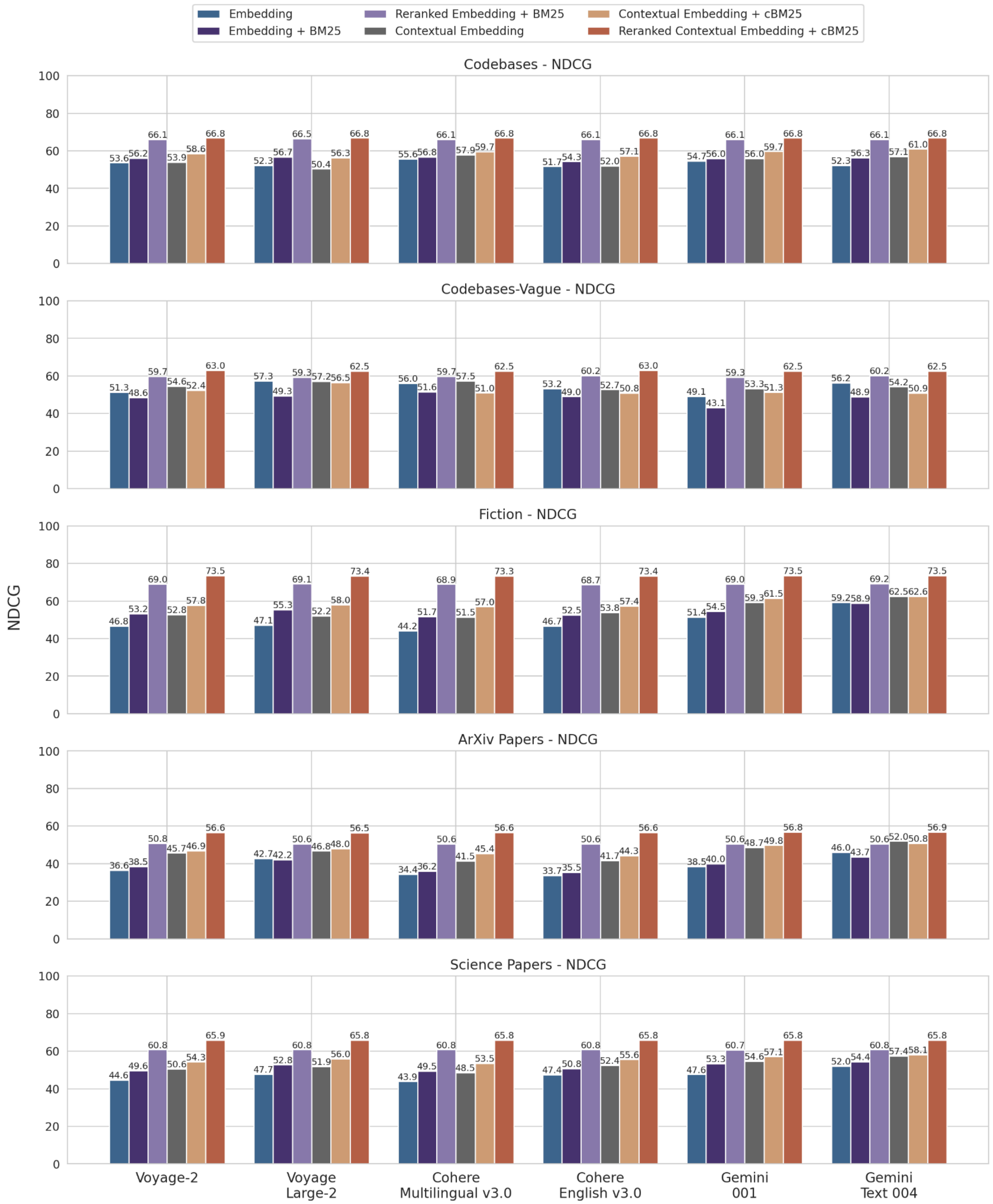
1 minus recall @ 10 results



1 minus recall @ 5 results



NDCG Results



Example Question & Answers from Evaluation Sets

Codebases

Query:
"What is the default layout used by BasicConfigurator if no layout is provided?"

Answer:
"If no layout is provided to `BasicConfigurator::configure(const LayoutPtr& layoutArg)`, it defaults to a PatternLayout with the conversion pattern `"%r [%t] %p %c %x - %m%n"`. This is defined by the static constant `TTCC_CONVERSION_PATTERN` in the code:

```cpp
static const helpers::WideLife<LogString>
TTCC_CONVERSION_PATTERN(LOG4CXX_STR("%r [%t] %p %c %x - %m%n"));
```
"

Golden Chunk:

```
"void BasicConfigurator::configure(const LayoutPtr& layoutArg)
{
	LogManager::getLoggerRepository()->setConfigured(true);
	auto layout = layoutArg;
	if (!layout)
	{
		static const helpers::WideLife<LogString>
TTCC_CONVERSION_PATTERN(LOG4CXX_STR("%r [%t] %p %c %x - %m%n"));
		layout = std::make_shared<PatternLayout>(TTCC_CONVERSION_PATTERN);
	}
	auto appender = std::make_shared<ConsoleAppender>(layout);
	Logger::getRootLogger()->addAppender(appender);
}

void BasicConfigurator::configure(const AppenderPtr& appender)
{
	LoggerPtr root = Logger::getRootLogger();
	root->addAppender(appender);
}

void BasicConfigurator::resetConfiguration()
{
	LogManager::resetConfiguration();
}
"
```

Context:

"This code snippet is part of the implementation of the `BasicConfigurator` class, which is part of the Log4cxx logging library for C++. The `BasicConfigurator` class provides a simple way to configure the logging system, including setting the layout and appender for the root logger. The `configure()` methods in this code snippet are responsible for performing this configuration, while the `resetConfiguration()` method is used to reset the logging configuration to its default state."

Codebases with Vague Questions

Question:
"How can I generate a vector of IPv4 addresses?"

Answer:
"You can use the `MakeIPv4s()` function to generate a vector of IPv4 addresses. For example:
```cpp
std::vector<in_addr> ipv4s = MakeIPv4s();
```
This function returns a vector with predefined IPv4 addresses:
```cpp
std::vector<in_addr> MakeIPv4s() {
    return {
        MakeIPv4(0x12345678), // 255.255.255.255
        MakeIPv4(0x0100007f), // 127.0.0.1
        MakeIPv4(3585395774),
        MakeIPv4(0),
        MakeIPv4(0x12345678),
    };
}
```
The IPv4 addresses are created using the `MakeIPv4` function, which takes an unsigned 32-bit integer as input."

Golden Chunk:

```
"std::vector<in_addr> MakeIPv4s() {
    return {
        MakeIPv4(0x12345678), // 255.255.255.255
        MakeIPv4(0x0100007f), // 127.0.0.1
        MakeIPv4(3585395774),
        MakeIPv4(0),
        MakeIPv4(0x12345678),
    };
}
```

```
std::vector<in6_addr> MakeIPv6s() {
   return {
      MakeIPv6(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15), //
1:203:405:607:809:a0b:c0d:e0f
      MakeIPv6(0, 0, 0, 0, 0, 1),                       // ::1
      MakeIPv6(0, 0, 0, 0, 0, 0),                       // ::
      MakeIPv6(0xff, 0xff, 204, 152, 189, 116),               // ::ffff:204.152.189.116
   };
}
"
```

Context:
" The provided code defines two functions, `MakeIPv4s()` and `MakeIPv6s()`, that generate vectors of IPv4 and IPv6 addresses, respectively. These functions are part of a larger set of value generator functions defined in the `value_generators.h` header file, which is included at the beginning of the document. The purpose of these value generator functions is to provide test data for various data types used in the Clickhouse database."

Fiction

Query:
"What are the initials of the mysterious man who met with Detective Inspector Greg Lestrade and offered to help him gain custody of his sons in exchange for information on Sherlock Holmes?"

Answer:
"The initials of the mysterious man who met with Detective Inspector Greg Lestrade are MH, as revealed by the signature at the end of the note he left for Lestrade."

Golden Chunk:
"The man withdrew a small notebook, flipping it open to a marked page, “Roger, age 6, fond of playing rugby and football, but could stand to focus more on his studies. And Aaron, age 3, a bit on the shy side, but loves to draw.”

“Are you _watching us?_ ” Greg stiffened at the thought of someone watching his sons, “You stay the _hell_ away from my sons!” He clenched his fists, “My problems are my own. And if you’re so good at watching people, why don’t you watch Sherlock on your own? Or better yet, talk to him your bloody self.” He hissed, turning red with anger, “Take me home. Now.”

The man watched him, like he was curious and amused by the spectacle, “Very well. The car will return you it picked you up. It was lovely speaking with you, Detective Inspector.”

Greg didn’t answer; he was shaking with anger at what had just transpired. What the hell _had_ happened? Did he just anger a mafia don? A corrupt politician? Was there going to be a hit on his head? Were his sons in danger because he lost his temper?

He couldn’t relax the entire time he drove to his house. He felt sick to his stomach with fear.

It was only when he opened the door to his home that he was able to relax slightly. Roger and Aaron were watching a movie on the couch while the nanny was cleaning up what must have been dinner. Sally, he noted, wasn’t home.

“I’m so sorry I’m late, Claire.” He smiled softly, grabbing a few extra bills, “Thank you for your time. I hope the boys didn’t give you too much trouble.”

Claire smiled, “Not at all, Mr. Lestrade. And your boss already came by the house to inform me that you would be late tonight.”

Greg blinked, “My boss?” He couldn’t imagine _anyone_ at work stopping by his house for any reason

The nanny nodded, “He left something for you on the table.”

He quickly thanked her and sent her on her way, his mind focused on what could possibly be on the table. His boys were so engrossed in the movie, they didn’t notice his nerves. With a shaky breath, Greg picked up the envelope that was on the table.

His name was written on it in elegant script. Inside was a plain white card, with the same script inside.

Don’t worry about your wife, Gregory. Consider it done. ~MH

End file.
"

Context
" This section depicts Greg Lestrade being confronted by a mysterious and powerful man, who seems to have intimate knowledge of Greg's personal life, including his marriage troubles and his children. The man offers to help Greg gain custody of his children in exchange for spying on Sherlock Holmes. Greg angrily refuses, and is then shocked when the man seems to have intervened in Greg's marital problems. This scene sets up the involvement of the mysterious "MH" character, who appears to have a vested interest in both Greg's and Sherlock's affairs."

Papers

Query:
"What theorem did White prove about the injectivity radius of closed hyperbolic 3-manifolds in terms of the rank of their fundamental group?"

Answer:
"White proved that for all r there is R such that every closed hyperbolic 3-manifold M with rank(π1(M)) ≤ r has inj(M) ≤ R. In other words, the injectivity radius of a closed hyperbolic 3-manifold is bounded above in terms of the rank of its fundamental group."

Golden Chunk:
"Let M be the infinite cover of M(F) corresponding to the kernel of π ; in the sequel we will always consider M with the unique hyperbolic metric such that the covering M → M(F) is Riemannian. Before going further we observe the following fact that we state here for further reference: Many of the arguments used in the present paper rely on properties of finitely generated subgroups of the fundamental group of M . Proposition 4.2 Every proper subgroup G of π 1 (M) π 1 (Σ g) of rank at most 2g is free and convex-cocompact.
Sketch of the proof
The manifold M is homeomorphic to Σ g ×R. In particular, every proper subgroup of π 1 (M) π 1 (Σ g) is either free or isomorphic to the fundamental group of a closed surface which covers Σ g with at least degree 2. Any such surface has genus greater than g and hence its fundamental group has rank greater than 2g. We have proved that the group G is free. A result due to Thurston in this case and to Agol [1] and Calegari-Gabai [4] in much more generality asserts that H 3 /G is homeomorphic to the interior of a handlebody. Now, Canary's generalization of Thurston's covering theorem [5] implies that G is convex-cocompact.
Proof of Theorem 1.1
As the kind reader may have deduced from the title of this section, we prove here Theorem 1.1. But first, as a warm-up, we show the result of White mentioned in the introduction:
Theorem (White [13]) For all r there is R such that every closed hyperbolic 3-manifold M with rank(π 1 (M)) ≤ r has inj(M) ≤ R.
Proof Let f : X → M be a minimal length carrier graph in the class of a minimal generating set of π 1 (M); observe that X has at most s = 3(r − 1) edges. Denote by X <t the (possibly empty) subgraph of X consisting of the union of all the edges with length less than t. Every simple closed circuit in X <t represents a non-trivial element in π 1 (M) by Lemma 2.2 and has at most length 3t(r − 1). In particular, it suffices to show that there is t r depending only on r such that some component Y of X <tr is not a tree.
Let l 0 be the constant provided by Lemma 3.1. Since M is closed we have that π 1 (M) is not free and in particularf :X → H 3 cannot be a quasi-isometric embedding. In particular, X <l 0 is not empty by Lemma 2.2 and Lemma 3.
is a (3(r − 1)l 0 , 3(r − 1)l 0)-quasi-isometric embedding. We obtain from Proposition 3.3 a constant l 1 = l 1 (r) depending only on r such that X <l 0 is a proper subgraph of X <l 1 . If again every connected component of X <l 1 is tree then we get l 2 = l 2 (r) depending only on r such

that X <l 1 is a proper subgraph of X <l 2 . This process can be repeated at most 3(r − 1) times since this is the number of edges in X ; this concludes the proof of White's Theorem.
As we see, the proof of White's Theorem yields in fact that every generating set (g 1 , . . . , g r) is Nielsen equivalent to a generating set (g 1 , . . . , g r) such that the translation length of g 1 is uniformly bounded. The idea of the proof of Theorem 1.1 is to show that every generating set of π 1 (M(F n)) is Nielsen equivalent to a generating set such that the translation lengths of all elements but 1 are uniformly bounded. "

Context:
"This section discusses the geometry of the infinite cover M of the mapping torus M(F), where F is a pseudo-Anosov mapping class on the closed surface Σ_g of genus g ≥ 2. The author establishes that every proper subgroup of π_1(M) of rank at most 2g is free and convex-cocompact. This geometric understanding of the fundamental group of M is then used in the subsequent proof of Theorem 1.1, which determines the rank of the fundamental group of the mapping tori M(F^n)."

ArXiv

Query:
"In the clustering algorithm experiments on the English Penn Treebank, how many models were used for each noise level σ?"

Golden Answer:
"For the clustering algorithm experiments on the English Penn Treebank, 20 models were used for each noise level σ ∈ { 0.05, 0.1, 0.15, 0.2 }, for a total of 80 models."

Golden Chunk:
"\section{Experiments}
\label{section:experiments}

In this section, we describe parsing experiments with two languages:
English and German.

\subsection{Results for English}

\begin{table*}[htb]
\begin{center}
{\small
\begin{tabular}{|c||c|c|c||c|c|c||c|c|c|}

```latex
\hline
& \multicolumn{3}{c||}{Clustering} & \multicolumn{3}{c||}{Spectral (smoothing)} &
\multicolumn{3}{c|}{Spectral (no smoothing)} \\
& MaxTre & MaxMrg & MaxEnt & MaxTre & MaxMrg & MaxEnt & MaxTre & MaxMrg & MaxEnt
\\
\hline
\setupAdd & 88.68 & 88.64 & 89.50 & 88.20 & 88.28 & 88.59 & 86.72 & 86.85 & 87.94 \\
\setupMul & 88.74 & 88.66 & 89.89  & 88.48 & 88.70 & 89.46 & 86.97 & 86.53 & 89.04 \\
\setupDropout & 88.68 & 88.56 & 89.80 & 88.64 & 88.71 & \textbf{89.47} & 88.37 & 88.06 &
89.52 \\
\setupAll & 88.84 & 88.75 & \textbf{89.95} & 88.38 & 88.75 & 89.45 & 87.49 & 87.00 &
\textbf{89.85} \\
\hline
No noise & \multicolumn{3}{c||}{86.48} & \multicolumn{3}{c||}{88.53 (Cohen et al., 2013)} &
\multicolumn{3}{c|}{86.47 (Cohen et al., 2013)} \\
\hline
\end{tabular}
}
\end{center}
\caption{\small Results on section 22 (WSJ). MaxTre denotes decoding
using maximal tree coverage, MaxMrg denotes decoding using maximal
marginal coverage, and MaxEnt denotes the use of a discriminative
reranker.  \setupAdd, \setupMul $\,$ and \setupDropout $\,$ denote
the use of additive Gaussian noise, multiplicative Gaussian noise
and dropout noise, respectively.  The number of models used in the
first three rows for the clustering algorithm is 80: 20 for each
$\sigma \in \{ 0.05, 0.1, 0.15, 0.2 \}$. For the spectral
algorithm, it is 20, 5 for each $\sigma$ (see footnotes).  The
number of latent states is $m=24$. For \setupAll, we use all
models combined from the first three rows.  The ``No noise''
baseline for the spectral algorithm is taken from Cohen et
al. (2013). The best figure in each algorithm block is in
boldface.\label{table:results-dev}}
\end{table*}
```

"

Context:

" This section describes the results of parsing experiments conducted by the authors on English and German language data. It presents detailed results on the English parsing task, including comparisons of different decoding techniques and the impact of using noisy models to improve performance. The results demonstrate that the authors' approach using a diverse set of predictions from multiple models outperforms previous state-of-the-art methods for English parsing."