CTA 분석 서버

기술보고서

㈜엑스큐어넷

승 인 내 역

사업자 : ㈜아울네스트

승인자 :	PM 고민수	(인)	**일자 :**	2025. .
검토자 :	QA	(인)	**일자 :**	2025. .

▣ 제·개정 이력

날짜	버전	작성자	승인자	내용
2025.12.01	v0.10	박상용	고민수	최초 작성
2025.12.12	v1.00	박상용	고민수	최종 확인

목 차

1 서론

1.1. 배경 및 목적

요약	현대전의 양상 변화 및 비정형 데이터 가치 증대
상세 내역	현대전의 양상이 물리적 타격 중심의 기동전에서 정보 우위 확보를 위한 인지전(Cognitive Warfare)과 정보전(Information Warfare)으로 급격히 전환됨에 따라, 군 내부 및 작전 환경에서 생성되는 비정형 데이터(Unstructured Data)의 전략적 가치는 그 어느 때보다 중요해지고 있다. 특히 국군방첩사령부(Defense Counterintelligence Command)를 포함한 국방 보안 기관에서 수집하는 방대한 통화 내역과 텍스트 데이터는 단순한 통신 기록을 넘어, 군 내 보안 위협 징후, 장병 사고 예방, 그리고 지휘관의 관심 사항을 조기에 식별할 수 있는 핵심 정보 자산으로 분류된다. 기존의 단순 키워드 매칭(Keyword Matching) 방식은 문맥을 파악하지 못해 과도한 오탐지(False Positive)를 유발하거나, 은어 및 변형된 군사 용어를 식별하지 못하는 미탐지(False Negative)의 한계를 노출하였다. 이에 따라, 자연어 처리(NLP, Natural Language Processing) 기술과 통계적 상관관계 분석(Statistical Correlation), 그리고 최신 딥러닝(Deep Learning) 모델을 융합하여 분석의 정확도와 효율성을 획기적으로 제고할 필요성이 대두되었다. 본 보고서는 현재 운용 중인 전군 통화내역 분석체계(CTA, Call Text Analytics)의 기술적 아키텍처, 데이터 처리 파이프라인, 그리고 핵심 분석 알고리즘을 국방 소프트웨어 기술문서 작성 가이드 및 관련 표준에 의거하여 심층적으로 기술한다. 본 문서는 체계의 유지보수 효율성 증대, 향후 고도화를 위한 기술적 베이스라인 확보, 그리고 유사 국방 정보체계 구축 시의 참조 모델 제공을 목적으로 작성되었다.

1.2. 범위 및 적용 대상

요약	기술보고서 분석 범위 및 주요 대상
상세 내역	본 보고서의 분석 범위는 CTA 체계의 전체 소프트웨어 수명 주기 중 상세 설계(Detailed Design) 및 구현(Implementation) 단계에 집중한다. 주요 분석 대상은 다음과 같다. 인프라 및 런타임 환경: 폐쇄망 환경에서의 고속 처리를 위한 운영체제(OS) 구성 및 이기종 언어(Python-C++) 통합 아키텍처. 데이터 처리 파이프라인: 수집, 전처리, 적재, 분석, 시각화로 이어지는 5단계 자동화 워크플로우. 핵심 알고리즘: C++ 기반 형태소 분석, Z-Score 기반 동시출현 네트워크, LDA 토픽 모델링, KoBART 생성형 요약. 엔지니어링 최적화: 병렬 처리(Multiprocessing), 메모리 관리(Memory Management), 데이터 직렬화(Serialization) 기술. 본 보고서는 국방 정보화 업무 훈령 및 무기체계 소프트웨어 개발 관리 매뉴얼을 준수하여 작성되었으며, 국방/안보 분야의 소프트웨어 엔지니어, 데이터 사이언티스트, 그리고 체계 운용 담당자를 주 독자층으로 상정한다.

2 체계 운용 환경 및 아키텍처

국방 정보체계는 민간 시스템과 달리 물리적 망 분리 환경(Closed Network)에서 운용되므로, 인터넷을 통한 외부 라이브러리 업데이트나 클라우드 자원 활용이 제한된다. 따라서 제한된 하드웨어 리소스 내에서 대용량 데이터를 실시간으로 처리하기 위한 고효율 아키텍처 설계가 필수적이다.

2.1. 하드웨어 및 운영체제 환경

- 본 체계는 안정성과 보안성이 검증된 엔터프라이즈급 리눅스 환경 위에서 구동된다.

운영체제	Rocky Linux 8.10 (Green Obsidian) - 선정 근거: RHEL(Red Hat Enterprise Linux)과 1:1 바이너리 호환성을 가지며, 국방 보안 가이드라인(STIG, Security Technical Implementation Guide)을 준수하는 보안 패치 적용이 용이하다.[1] 특히 대용량 텍스트 로그의 생성과 삭제가 빈번한 분석 서버의 특성을 고려하여, 커널 레벨에서의 I/O 스케줄러 최적화 및 파일 시스템 튜닝이 적용되었다. 이는 시스템의 신뢰성(Reliability)과 가용성(Availability)을 보장하기 위한 핵심 조치이다.
데이터베이스 (DBMS)	PostgreSQL Cluster – 이원화 구조: 데이터의 수집(Insert/Update) 트랜잭션과 분석(Select/Aggregation) 트랜잭션 간의 자원 경합(Resource Contention)을 방지하기 위해, ODS(Operational Data Store)와 DW(Data Warehouse)로 물리적/논리적으로 분리된 클러스터를 운용한다. – ODS: 실시간 통화 내역 적재 및 1차 가공 데이터 저장. – DW: 형태소 분석 결과, 통계 데이터, 분석 모델링 결과 등 장기 보관 및 분석용 데이터 저장. – 연결 관리: Python의 psycopg2 라이브러리를 활용하여 DB 커넥션 풀링(Connection Pooling)을 구현, 다수 프로세스의 동시 접근 시에도 안정적인 세션을 유지한다.

2.2. 하이브리드 런타임 아키텍처

단일 프로그래밍 언어의 성능적 한계를 극복하고, 각 언어의 장점을 결합하기 위해 Python과 C++를 융합한 하이브리드 아키텍처를 채택하였다. 이는 국방 분야의 고성능 컴퓨팅(HPC) 요구사항과 개발 생산성을 동시에 만족시키기 위한 전략적 설계이다.

표 1. CTA 체계 하이브리드 런타임 구성 및 역할

구성 요소	언어 및 버전	주요 역할 및 기술적 특징	선정 사유
Control Plane	Python 3.11	전체 파이프라인 오케스트레이션, 최신 Deep Learning 모델(KoBART) 구동, DB 핸들링 (psycopg2), 시각화 전처리	풍부한 라이브러리 생태계(Pandas, PyTorch 등) 활용 및 개발 생산성 확보.
Computational Core	C++ (Native)	초고속 형태소 분석 엔진(NLPMF), 메모리 집약적 텍스트 전처리, 사전(Dictionary) 관리	Python GIL(Global Interpreter Lock) 회피, 포인터 연산을 통한 메모리 직접 제어로 처리 속도 극대화.
Legacy Support	Python 2.7	기존 통계 분석 모듈(cooccurrence_matrix2_cta.py), 레거시 토픽 모델링 엔진 호환성 유지	기 검증된 분석 자산의 재사용성 보장 및 시스템 마이그레이션 리스크 최소화.

2.2.1 FFI(Foreign Function Interface) 기반 통합

이 기종 언어 간의 통신은 ctypes 라이브러리를 통해 구현되었다. 일반적인 소켓 통신이나 파일 기반 I/O 방식은 직렬화/역직렬화(Serialization/Deserialization) 과정에서 상당한 오버헤드를 발생시킨다. 반면, 본 체계는 C++로 컴파일된 공유 라이브러리(libnlpmf.so.2)를 Python 프로세스의 메모리 공간에 직접 로드(ctypes.cdll.LoadLibrary)하여 함수를 호출한다.

이 방식은 Python 객체와 C++ 메모리 구조체 간의 직접 매핑을 가능하게 하여, 대용량 텍스트 데이터를 복사 없이 처리(Zero-copy processing)할 수 있게 한다. 이는 분석 속도를 수십 배 향상시키는 핵심 엔지니어링 기술이다.

3 데이터 처리 파이프라인 상세 분석

- 본 체계의 데이터 처리는 [사전 구축] → [수집/전처리] → [심층 분석] → [시각화/요약]의 전 과정을 5단계의 정교한 자동화 배치(Batch)로 수행한다. 각 단계는 데이터의 무결성(Integrity)과 처리의 완결성을 보장하도록 설계되었다.

3.1. 도메인 특화 사전 자동 구축

– 국방 데이터는 일반 사회 용어와 다른 특수성(부대명, 작전 용어, 약어 등)을 가진다. 이를 정확히 분석하기 위해 사용자 사전(User Dictionary)을 매일 자동으로 갱신하고 컴파일한다.

관련 모듈	01_make_cta_custom_dict.py, dict_maker (Shell Script).
구현 절차	1. 키워드 추출: std.intrst_kwd_info 테이블에서 최신 관심 키워드 리스트를 조회한다. 2. 비용(Cost) 엔지니어링: 형태소 분석기는 Viterbi 알고리즘을 통해 가장 확률이 높은 형태소 조합 경로를 찾는다. 이때 군사 용어가 일반 명사보다 우선순위를 갖도록 엔진 내부 파라미터인 비용(Cost)을 강제로 낮게 설정한다. - 설정 값: Cost 1783 또는 1784, 식별자 3538 또는 3541, 품사 태그 NNG (일반 명사). - 효과: "작전/NNG"과 같은 군사 용어가 복합 명사 분해 과정에서 우선적으로 선택되도록 유도한다. 3. 바이너리 컴파일: 텍스트 기반의 CSV 사전 파일을 메모리 로딩 속도가 최적화된 바이너리 포맷(.dic)으로 컴파일하여 배포한다.

3.2. 고성능 전처리 및 데이터 적재

– 수집된 비정형 텍스트 데이터를 분석 가능한 형태소 단위로 분해하고, 핵심 명사를 추출하여 DW에 적재하는 단계이다.

관련 모듈	02_insert_dw.talk_cn_info_every5min_nlp_step0.py.
핵심 기술 및 로직	델타 로딩(Delta Loading): 데이터베이스 부하를 최소화하기 위해 전체 데이터를 재처리하지 않고, WHERE DATE(cmi.ctime_start) = '{yesterdayStr}' 조건을 통해 전일 생성된 데이터만 선별적으로 처리한다. 트랜잭션 안전성(Transactional Safety): 재작업 시 데이터 중복을 방지하기 위해, 적재 전 해당 일자의 데이터를 선삭제(Delete)하고 후삽입(Insert)하는 멱등성(Idempotency) 로직을 적용하였다. C++ 엔진 바인딩: mtagger = NLPMF.Tagger('-d...')를 통해 C++ 엔진 인스턴스를 생성하고, mtagger.parse(sentence) 함수를 호출하여 초고속 형태소 분석을 수행한다. 복합명사 처리 (NPMaker): '삼성페이'와 같이 하나의 의미를 가지는 복합 명사가 '삼성'과 '페이'로 분리되는 것을 방지하기 위해, NPMaker 함수를 통해 인접한 명사를 결합하는 후처리 로직을 수행한다. 이는 분석의 의미적 정확도(Semantic Accuracy)를 높이는 중요한 과정이다. 불용어(Stopword) 필터링: std.stopword_info에 등록된 불용어를 메모리상에서 필터링하여 분석 노이즈를 제거한다.

3.3. 분석 오케스트레이션

- 데이터 처리의 단계별 종속성을 관리하고, 병렬 처리를 제어하는 단계이다.

관련 모듈	03_R04_manual.py, CoreAnalyticsCTA.py.
구현 특징	절차적 제어(Procedural Control): os.system 호출의 반환값을 확인하여 선행 작업(Step 02)이 성공적으로 완료된 경우에만 후행 분석(Step 04, 05)을 트리거한다. Job Context 관리: uuid.uuid5를 사용하여 분석 작업마다 고유 식별자(Job ID)를 생성한다. 이를 통해 전체 분석 파이프라인의 트랜잭션 로그를 추적하고, 장애 발생 시 원인 분석을 용이하게 한다. 부대별 파티셔닝: std.troop_fn_info 테이블의 부대 코드(troop_cd)를 기반으로 반복 루프를 실행하여, 각 부대별로 독립적인 분석 컨텍스트를 생성한다.

3.4. 네트워크 매트릭스 연산

- 단어 간의 동시 출현 빈도를 정량적으로 계산하여 네트워크 그래프 구조로 변환하는 단계이다. 가장 많은 연산 리소스를 소모하는 단계이기도 하다.

관련 모듈	04_run.py, cooccurrence_matrix2_cta.py.
주요 알고리즘 및 기술	병렬 처리(Multiprocessing): multiprocessing.cpu_count()를 통해 시스템의 가용 CPU 코어 수를 확인하고, 전체 텍스트 데이터를 균등한 청크(Chunk)로 분할하여 병렬 연산을 수행한다. 각 프로세스는 독립적으로 교집합 연산을 수행하고, 결과는 cPickle로 직렬화되어 저장된다. Z-Test 기반 필터링: 단순 빈도가 아닌 통계적 유의성을 검증하기 위해 Z-Score를 산출한다. (상세 내용은 4.1절 참조) Pajek 포맷 변환: 네트워크 분석 도구인 Pajek에서 사용하는 .net 파일 포맷을 생성하고, 이를 다시 웹 시각화를 위해 JSON 포맷으로 변환(net2json.py)한다. 지터링(Jittering): 시각화 시 노드(단어) 간의 좌표 겹침을 방지하기 위해, 산출된 X, Y 좌표에 random.uniform(0.1, 0.25) 함수를 이용한 미세한 난수 변위를 적용한다.

3.5. 생성형 요약 및 토픽 모델링

- 분석 결과를 사용자가 직관적으로 이해할 수 있도록 요약문과 주요 토픽을 생성한다.

관련 모듈	05_getSummarization.py, getTopic.py.
기술적 특징	KoBART 적용: 한국어 처리에 특화된 KoBART (Korean BART) 모델을 transformers 라이브러리를 통해 로드한다. GPU 가속: os.environ = '1' 설정을 통해 CUDA 커널의 동기화 실행을 제어하고, PyTorch 기반의 GPU 가속 추론을 수행하여 대량의 텍스트 요약 속도를 확보한다.[1] 데이터 정제: re.sub 정규표현식을 사용하여 "통신보안", "충성" 등 요약 품질을 저해하는 군사 상투어를 전처리 단계에서 제거한다.

4 핵심 분석 알고리즘 심층 분석

- CTA 체계의 분석 품질은 적용된 수학적 모델과 알고리즘의 정교함에 기인한다. 본 장에서는 시스템에 적용된 핵심 알고리즘의 이론적 배경과 구현 상세를 다룬다.

4.1. 고성능 동시출현 네트워크

- 단어 간의 관계를 규명하기 위해 동시출현(Co-occurrence) 빈도를 기반으로 네트워크를 구성한다. 이는 "함께 자주 등장하는 단어는 의미적으로 밀접한 관계가 있다"는 분포 가설(Distributional Hypothesis)에 근거한다.

4.1.1. 수학적 정의 및 Z-Score 검정

상세 내역	단순히 빈도가 높은 단어 쌍(예: '군인'-'부대')은 정보가치가 낮을 수 있다. 따라서 우연에 의한 동시 출현을 배제하고 통계적으로 유의미한 관계(Significant Collocation)만을 추출하기 위해 Z-Score 검정을 수행한다. 두 단어 w_i, w_j의 관측된 동시 출현 빈도를 O_{ij}, 전체 코퍼스에서의 기대 빈도를 E_{ij}라고 할 때, Z-Score Z_{ij}는 다음과 같이 정의된다. $$Z_{ij} = \frac{O_{ij} - E_{ij}}{\sigma_{ij}}$$ 여기서 σ_{ij}는 표준편차이다. 본 체계는 이 Z 값을 정규분포의 누적분포함수(CDF, Cumulative Distribution Function)에 대입하여 확률값(P-value)을 산출한다. 구현 로직: 각 단어 쌍의 빈도 분포에 대해 평균(μ)과 표준편차(σ)를 계산한다. $Z = \frac{r - \mu}{\sigma}$ 공식을 통해 표준화된 점수를 획득한다. nlpstat.normcdf(z) 함수를 통해 P-value를 계산하고, 신뢰수준 95% ($Z > 1.96$) 이상인 연결선(Edge)만을 네트워크에 남긴다. 이러한 통계적 필터링은 노이즈를 제거하고, 보안 사고나 특이 징후와 같이 "평소와 다르게 강하게 연결된" 키워드 쌍을 탐지하는 데 탁월한 성능을 발휘한다.

4.1.2. 대칭 행렬 및 희소성 처리

상세 내역	단어 간의 관계는 방향성이 없으므로(Undirected), 인접 행렬(Adjacency Matrix)은 대칭 행렬($M_{ij} = M_{ji}$)의 형태를 띤다. 시스템은 메모리 효율성을 위해 상삼각 행렬(Upper Triangular Matrix) 성분만 계산하고, 대칭성을 이용하여 하삼각 성분을 복제하는 방식을 사용한다. 또한, 전체 단어 조합 중 실제 연결이 존재하는 비율이 낮은 희소 행렬(Sparse Matrix)의 특성을 고려하여, 0이 아닌 값만 저장하는 dict 자료 구조를 활용해 메모리 사용량을 최적화하였다.

4.2. 잠재 디리클레 할당(LDA) 기반 토픽 모델링

- 비정형 텍스트 집합에서 숨겨진 주제(Latent Topic)를 확률적으로 추론하기 위해 LDA(Latent Dirichlet Allocation) 알고리즘을 적용하였다.

4.2.1. 하이퍼파라미터(Hyperparameter) 최적화

- LDA 모델의 성능은 사전 분포(Prior Distribution)를 결정하는 하이퍼파라미터 α와 β에 크게 의존한다. 본 체계는 군 통화 데이터의 특성(명확한 용건, 짧은 대화)을 반영하여 파라미터를 튜닝하였다.

표 2. LDA 하이퍼파라미터 설정 및 기술적 근거

파라미터	설정값	의미 및 설정 근거
Alpha (α)	0.1	문서 내 토픽 분포의 희소성(Sparsity) 제어. 값을 1.0 미만으로 낮게 설정하여, 하나의 통화 문서가 소수의 명확한 주제(예: '휴가', '훈련' 중 하나)에 집중되도록 유도함.
Beta (β)	0.001	토픽 내 단어 분포의 희소성 제어. 매우 낮은 값으로 설정하여, 각 토픽을 구성하는 단어들이 뚜렷하게 구별되도록 함. 이는 모호한 토픽 생성을 방지하고 분석의 해석 용이성을 높임.

Topic Count (K)	100	전군 차원의 다양한 이슈(보안, 인사, 군수, 작전 등)를 포괄하기 위해 클러스터 개수를 100개로 설정하여 세밀한 분류가 가능하도록 함.

4.2.2. 보안 파티셔닝 모델링

상세 내역	보안상의 이유로, 부대 간의 민감한 정보가 혼재되는 것을 방지하기 위해 부대 코드(troop_cd)별로 데이터를 파티셔닝(Partitioning)하여 독립적인 LDA 모델링을 수행한다. 이는 물리적으로 분리된 데이터셋에 대해 별도의 모델 인스턴스를 생성하는 방식으로 구현되어, 정보 유출 가능성을 원천 차단한다.

4.3. KoBART 기반 생성형 요약

- 추출적 요약(Extractive Summarization)의 문맥 연결 부자연스러움을 극복하기 위해, Transformer 아키텍처 기반의 생성형 요약 모델인 KoBART를 도입하였다.

상세 내역	- BPE (Byte Pair Encoding): 한국어의 교착어적 특성을 반영하여, 단어보다 작은 하위 단어(Subword) 단위로 토큰화를 수행한다. 이는 미등록 단어(OOV, Out-Of-Vocabulary) 문제에 강건하게 대응할 수 있게 한다. - Encoder-Decoder 구조: 입력 텍스트의 문맥을 양방향으로 이해하는 인코더(Encoder)와, 요약문을 순차적으로 생성하는 디코더(Decoder) 구조를 통해 유창하고 자연스러운 한국어 요약문을 생성한다.

5 엔지니어링 최적화

- 대규모 국방 데이터를 제한된 시간 내에 처리하기 위해 적용된 주요 엔지니어링 기술은 다음과 같다.

5.1. 고속 데이터 직렬화

- Python의 프로세스 간 통신(IPC)에서 발생하는 병목을 해소하기 위해 cPickle 모듈을 사용한다. 특히 HIGHEST_PROTOCOL 옵션을 적용하여 데이터를 바이너리 포맷으로 직렬화함으로써, 텍스트 기반 포맷(JSON/XML) 대비 수십 배 빠른 I/O 성능을 달성하였다. 이는 멀티프로세싱 환경에서 부모-자식 프로세스 간의 대용량 데이터 교환 시 지연 시간을 최소화하는 핵심 요소이다.

5.2. 멀티프로세싱을 통한 병렬성 극대화

- 단일 스레드로 동작하는 Python의 GIL 제약을 극복하기 위해 multiprocessing 모듈을 활용한 데이터 병렬화(Data Parallelism)를 구현하였다.

상세내역	Chunking Strategy: 전체 데이터 리스트를 CPU 코어 수(N)로 등분하여 N개의 자식 프로세스에 할당한다. Load Balancing: 각 프로세스에 할당되는 데이터 양을 균등하게 조절하여 특정 코어에 부하가 집중되는 것을 방지한다. Result Aggregation: 각 프로세스의 부분 연산 결과(.dict 파일)를 메인 프로세스가 수집하여 최종 병합(Merge)하는 Map-Reduce 방식을 적용하였다.

5.3. 데이터베이스 트랜잭션 최적화

- 대량의 데이터를 DW에 적재(COPY or INSERT)하는 과정에서 발생할 수 있는 성능 저하를 방지하기 위해, psycopg2의 executemany 기능을 활용하여 배치 처리를 수행한다. 또한, 인덱스(Index) 업데이트로 인한 쓰기 지연을 최소화하기 위해 대량 적재 시 일시적으로 인덱스를 비활성화하거나, 파티셔닝 테이블(Partitioned Table)을 활용하여 관리 효율성을 높였다.

6 결론 및 기대효과

- 본 보고서에서 분석한 전군 통화내역 분석체계(CTA)는 국방 분야의 비정형 빅데이터 분석 역량을 한 단계 도약시킨 사례로 평가된다.

1. 분석의 신뢰성 및 정밀도 확보: 단순 빈도 분석을 넘어 Z-Score 검정과 정규분포 모델링을 적용함으로써, 통계적으로 유의미한 보안 위협 징후만을 정밀하게 탐지할 수 있는 기반을 마련하였다. 또한, LDA 파라미터 튜닝을 통해 군 특수 용어와 주제를 명확히 분류함으로써 분석 결과의 가독성과 활용도를 높였다.
2. 고성능 하이브리드 아키텍처 구현: Python의 생산성과 C++의 고성능을 결합하고, GPU 가속 및 병렬처리 기술을 접목하여 폐쇄망 환경의 리소스 제약을 극복하고 대용량 데이터의 실시간 처리를 가능케 하였다. 이는 향후 타 국방 정보체계 구축 시 참조할 수 있는 최적화된 아키텍처 모델을 제시한다.
3. 지능형 의사결정 지원 체계 확립: KoBART 기반의 자동 요약 기술을 통해 지휘관 및 분석관이 방대한 데이터 속에서 핵심 정보를 즉각적으로 파악하고, 선제적인 조치를 취할 수 있도록 지원함으로써 국방 보안 태세 강화에 직접적으로 기여한다.

향후 본 체계는 음성 인식(STT) 성능의 고도화, 실시간 스트리밍 데이터 분석 파이프라인 통합, 그리고 거대언어모델(LLM)의 온프레미스(On-Premise) 구축을 통해, 국방 AI 전력의 중추적인 역할을 수행할 것으로 기대된다.

Reference

1. 무기체계 소프트웨어 개발 및 관리 매뉴얼 – 방위사업청, accessed on December 12, 2025, https://www.dapa.go.kr/jfile/board/readDownloadFile.do?fileType=WT_CMS_DOC&fileTypeSeq=16277&fileNum=1
2. Understanding and Achieving Software Reliability | www.dau.edu, accessed on December 12, 2025, https://www.dau.edu/acquipedia-article/understanding-and-achieving-software-reliability
3. U.S. Department of Defense releases security configuration standard for Red Hat Enterprise Linux 7, accessed on December 12, 2025, https://www.redhat.com/es/blog/us-department-defense-releases-security-configuration-standard-red-hat-enterprise-linux-7
4. C++ vs. Python: Full Comparison and Insights | Capaciteam Blog, accessed on December 12, 2025, https://capaciteam.com/c-plus-plus-vs-python/
5. Running C/C++ Code within Python: Bridging Efficiency and Flexibility, accessed on December 12, 2025, https://mostafa-amin.com/post/running-cc-code-within-python-bridging-efficiency-and-flexibility/
6. C++ VS Python for Machine Learning- Elinext Blog, accessed on December 12, 2025, https://www.elinext.com/solutions/ai/machine-learning/trends/cpp-vs-python-for-machine-learning/
7. Python vs. C++: Execution Performance and Development Efficiency, accessed on December 12, 2025, https://www.sencury.com/post/python-vs-c-execution-performance-and-development-efficiency
8. ctypes — A foreign function library for Python — Python 3.14.2 documentation, accessed on December 12, 2025, https://docs.python.org/3/library/ctypes.html
9. If you're using Python for performance-critical applications, simple use of the ... - Hacker News, accessed on December 12, 2025, https://news.ycombinator.com/item?id=7231242
10. 네이버 블로그로 동시출현단어 분석 (co-occurence network anlysis), accessed on December 12, 2025, https://junhewk.github.io/text/2017/08/08/cooccurence-matrix-with-Naver-blog/
11. Survey of Word Co-occurrence Measures for Collocation Detection, accessed on December 12, 2025, https://www.scielo.org.mx/pdf/cys/v20n3/1405-5546-cys-20-03-00327.pdf
12. Z-Score: Definition, Formula, Calculation & Interpretation - Simply Psychology, accessed on December 12, 2025, https://www.simplypsychology.org/z-score.html
13. Unit 6 Making statistical claims, accessed on December 12, 2025, https://www.lancaster.ac.uk/fass/projects/corpus/ZJU/xCBLS/chapters/A06.pdf
14. Scalable Hyperparameter Selection for Latent Dirichlet Allocation - Statistics - University of Florida, accessed on December 12, 2025, https://users.stat.ufl.edu/~doss/Research/shs-lda.pdf
15. Dirichlet distributions, the alpha hyperparameter, and LDA - AmCAT, accessed on December 12, 2025, https://i.amcat.nl/lda/understanding_alpha.html