

hizkuntza

Language



**Language  
technology,  
the next frontier  
of Artificial  
Intelligence**

Annual Report  
2021

# INTRODUCTION



words

language

## hizkuntza

Language understanding is a main goal of artificial intelligence and, although much progress has been made in deep learning, it remains one of the main limitations of artificial intelligence because computers will not attain intelligence until they understand language. In order to face the new challenges and opportunities of artificial intelligence and the digital transformation, it is essential to foster research in language technologies. To this end, we created HiTZ with a dual purpose: to be an international referent in language technology research and to be the referent in the computational treatment of Basque. The center engages in five lines of action: training, research, innovation and transfer, fostering our culture and societal

center

```

def __str__(self):
    """Return a string representation of the object"""
    return f"Team {self.id} with {self.name} members"

def __repr__(self):
    """Return a string representation of the object"""
    return f"Team(id={self.id}, name={self.name}, members={self.members})"

def __eq__(self, other):
    """Return True if the object is equal to the other object"""
    return self.id == other.id and self.name == other.name

def __lt__(self, other):
    """Return True if the object is less than the other object"""
    return self.id < other.id

def __gt__(self, other):
    """Return True if the object is greater than the other object"""
    return self.id > other.id

def __le__(self, other):
    """Return True if the object is less than or equal to the other object"""
    return self.id <= other.id

def __ge__(self, other):
    """Return True if the object is greater than or equal to the other object"""
    return self.id >= other.id

def __hash__(self):
    """Return the hash value of the object"""
    return hash((self.id, self.name))

def __getitem__(self, index):
    """Return the member at the given index"""
    return self.members[index]

def __setitem__(self, index, member):
    """Set the member at the given index"""
    self.members[index] = member

def __delitem__(self, index):
    """Delete the member at the given index"""
    del self.members[index]

def __iter__(self):
    """Return an iterator over the members"""
    return iter(self.members)

def __len__(self):
    """Return the number of members"""
    return len(self.members)

def __contains__(self, member):
    """Return True if the member is in the team"""
    return member in self.members

def __add__(self, other):
    """Return the union of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members + other.members)

def __sub__(self, other):
    """Return the difference of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members - other.members)

def __mul__(self, other):
    """Return the product of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members * other.members)

def __div__(self, other):
    """Return the quotient of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members / other.members)

def __mod__(self, other):
    """Return the remainder of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members % other.members)

def __pow__(self, other):
    """Return the power of the two teams"""
    return Team(id=self.id, name=self.name, members=self.members ** other.members)

def __divmod__(self, other):
    """Return the quotient and remainder of the two teams"""
    return (self / other, self % other)

def __radd__(self, other):
    """Return the sum of the two teams"""
    return other + self

def __rsub__(self, other):
    """Return the difference of the two teams"""
    return other - self

def __rmul__(self, other):
    """Return the product of the two teams"""
    return other * self

def __rdiv__(self, other):
    """Return the quotient of the two teams"""
    return other / self

def __rmod__(self, other):
    """Return the remainder of the two teams"""
    return other % self

def __rpow__(self, other):
    """Return the power of the two teams"""
    return other ** self

def __iadd__(self, other):
    """Return the sum of the two teams"""
    self.members += other.members
    return self

def __isub__(self, other):
    """Return the difference of the two teams"""
    self.members -= other.members
    return self

def __imul__(self, other):
    """Return the product of the two teams"""
    self.members *= other.members
    return self

def __idiv__(self, other):
    """Return the quotient of the two teams"""
    self.members /= other.members
    return self

def __imod__(self, other):
    """Return the remainder of the two teams"""
    self.members %= other.members
    return self

def __ipow__(self, other):
    """Return the power of the two teams"""
    self.members **= other.members
    return self

def __iand__(self, other):
    """Return the intersection of the two teams"""
    self.members = self.members & other.members
    return self

def __ior__(self, other):
    """Return the union of the two teams"""
    self.members = self.members | other.members
    return self

def __ixor__(self, other):
    """Return the symmetric difference of the two teams"""
    self.members = self.members ^ other.members
    return self

def __iand__(self, other):
    """Return the intersection of the two teams"""
    self.members = self.members & other.members
    return self

def __ior__(self, other):
    """Return the union of the two teams"""
    self.members = self.members | other.members
    return self

def __ixor__(self, other):
    """Return the symmetric difference of the two teams"""
    self.members = self.members ^ other.members
    return self

```

```

class Team:
    """A class representing a team"""

    def __init__(self, id, name, members):
        self.id = id
        self.name = name
        self.members = members

    def __str__(self):
        """Return a string representation of the object"""
        return f"Team {self.id} with {self.name} members"

    def __repr__(self):
        """Return a string representation of the object"""
        return f"Team(id={self.id}, name={self.name}, members={self.members})"

    def __eq__(self, other):
        """Return True if the object is equal to the other object"""
        return self.id == other.id and self.name == other.name

    def __lt__(self, other):
        """Return True if the object is less than the other object"""
        return self.id < other.id

    def __gt__(self, other):
        """Return True if the object is greater than the other object"""
        return self.id > other.id

    def __le__(self, other):
        """Return True if the object is less than or equal to the other object"""
        return self.id <= other.id

    def __ge__(self, other):
        """Return True if the object is greater than or equal to the other object"""
        return self.id >= other.id

    def __hash__(self):
        """Return the hash value of the object"""
        return hash((self.id, self.name))

    def __getitem__(self, index):
        """Return the member at the given index"""
        return self.members[index]

    def __setitem__(self, index, member):
        """Set the member at the given index"""
        self.members[index] = member

    def __delitem__(self, index):
        """Delete the member at the given index"""
        del self.members[index]

    def __iter__(self):
        """Return an iterator over the members"""
        return iter(self.members)

    def __len__(self):
        """Return the number of members"""
        return len(self.members)

    def __contains__(self, member):
        """Return True if the member is in the team"""
        return member in self.members

    def __add__(self, other):
        """Return the union of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members + other.members)

    def __sub__(self, other):
        """Return the difference of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members - other.members)

    def __mul__(self, other):
        """Return the product of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members * other.members)

    def __div__(self, other):
        """Return the quotient of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members / other.members)

    def __mod__(self, other):
        """Return the remainder of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members % other.members)

    def __pow__(self, other):
        """Return the power of the two teams"""
        return Team(id=self.id, name=self.name, members=self.members ** other.members)

    def __divmod__(self, other):
        """Return the quotient and remainder of the two teams"""
        return (self / other, self % other)

    def __radd__(self, other):
        """Return the sum of the two teams"""
        return other + self

    def __rsub__(self, other):
        """Return the difference of the two teams"""
        return other - self

    def __rmul__(self, other):
        """Return the product of the two teams"""
        return other * self

    def __rdiv__(self, other):
        """Return the quotient of the two teams"""
        return other / self

    def __rmod__(self, other):
        """Return the remainder of the two teams"""
        return other % self

    def __rpow__(self, other):
        """Return the power of the two teams"""
        return other ** self

    def __iadd__(self, other):
        """Return the sum of the two teams"""
        self.members += other.members
        return self

    def __isub__(self, other):
        """Return the difference of the two teams"""
        self.members -= other.members
        return self

    def __imul__(self, other):
        """Return the product of the two teams"""
        self.members *= other.members
        return self

    def __idiv__(self, other):
        """Return the quotient of the two teams"""
        self.members /= other.members
        return self

    def __imod__(self, other):
        """Return the remainder of the two teams"""
        self.members %= other.members
        return self

    def __ipow__(self, other):
        """Return the power of the two teams"""
        self.members **= other.members
        return self

    def __iand__(self, other):
        """Return the intersection of the two teams"""
        self.members = self.members & other.members
        return self

    def __ior__(self, other):
        """Return the union of the two teams"""
        self.members = self.members | other.members
        return self

    def __ixor__(self, other):
        """Return the symmetric difference of the two teams"""
        self.members = self.members ^ other.members
        return self

    def __iand__(self, other):
        """Return the intersection of the two teams"""
        self.members = self.members & other.members
        return self

    def __ior__(self, other):
        """Return the union of the two teams"""
        self.members = self.members | other.members
        return self

    def __ixor__(self, other):
        """Return the symmetric difference of the two teams"""
        self.members = self.members ^ other.members
        return self

```

```
RESULT_TYPES = ["visual"]
```

```
def members
```

```
team1.members += team2
```

```
end
```

```
def __str__(self):
```

```
return f"Team {self.id} with {self.name} members"
```

```
def __repr__(self):
```

```
return f"Team(id={self.id}, name={self.name}, members={self.members})"
```

```
def __eq__(self, other):
```

```
return self.id == other.id and self.name == other.name
```

```
def __lt__(self, other):
```

```
return self.id < other.id
```

```
def __gt__(self, other):
```

```
return self.id > other.id
```

```
def __le__(self, other):
```

```
return self.id <= other.id
```

```
def __ge__(self, other):
```

```
return self.id >= other.id
```

```
def __hash__(self):
```

```
return hash((self.id, self.name))
```

```
def __getitem__(self, index):
```

```
return self.members[index]
```

```
def __setitem__(self, index, member):
```

```
self.members[index] = member
```

```
def __delitem__(self, index):
```

```
del self.members[index]
```

```
def __iter__(self):
```

```
return iter(self.members)
```

```
def __len__(self):
```

```
return len(self.members)
```

```
def __contains__(self, member):
```

```
return member in self.members
```

```
def __add__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members + other.members)
```

```
def __sub__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members - other.members)
```

```
def __mul__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members * other.members)
```

```
def __div__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members / other.members)
```

```
def __mod__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members % other.members)
```

```
def __pow__(self, other):
```

```
return Team(id=self.id, name=self.name, members=self.members ** other.members)
```

```
def __divmod__(self, other):
```

```
return (self / other, self % other)
```

```
def __radd__(self, other):
```

```
return other + self
```

```
def __rsub__(self, other):
```

```
return other - self
```

```
def __rmul__(self, other):
```

```
return other * self
```

```
def __rdiv__(self, other):
```

```
return other / self
```

```
def __rmod__(self, other):
```

```
return other % self
```

```
def __rpow__(self, other):
```

```
return other ** self
```

```
def __iadd__(self, other):
```

```
self.members += other.members
```

technology

impact. This wide scope requires a broad interdisciplinary team, which brings together computer scientists, linguists, engineers, translators and sociologists, among others. The work of the researchers from Aholab and Ixa, the two research groups that make up the HiTZ center, bore fruit this year. Of note were the successful creation of the first neural language models for Basque, leadership in the European Language Equality project that seeks to define a strategy for comprehensive access to digital language technology across Europe, advances in the European research infrastructure CLARIN, and five awards received at the most prominent international congresses and in various scientific competitions, including one or-

ganized by the United States government in response to the Covid-19 pandemic. Most notably, we received two National Research Awards in Computer Science and our director was recognised as one of the 15 fellows that our main research association has in Europe. Finally, one of our PhDs was awarded for the best thesis in Artificial Intelligence in Europe (EurAI). This solid foundation allows us to continue with renewed energy and, in cooperation with technology centers, companies and institutions, turn the Basque Country into an international hub for language technologies and artificial intelligence.

Eneko Agirre (Director of HiTZ) and German Rigau (Deputy Director of HiTZ)

words

words

hizkuntza

# HiTZ IN NUMBERS

## Research & Transfer

# 34

Projects

# 22

Transfer Contracts

# 65

Publications

4

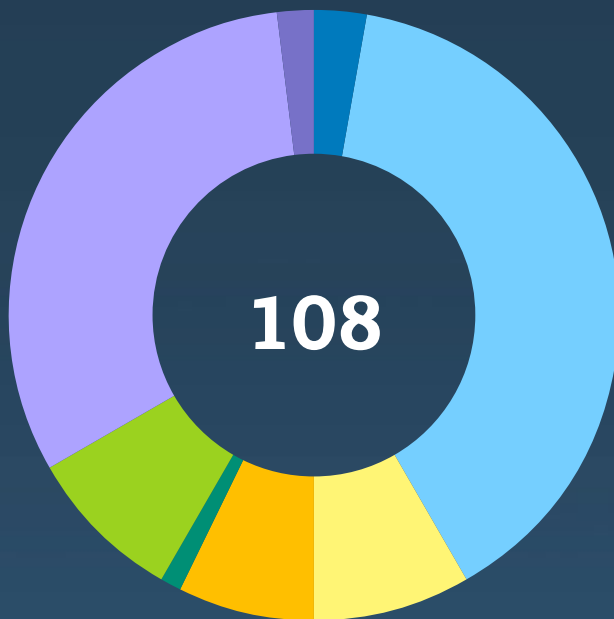
BASQUE CENTER FOR LANGUAGE TECHNOLOGY

techn

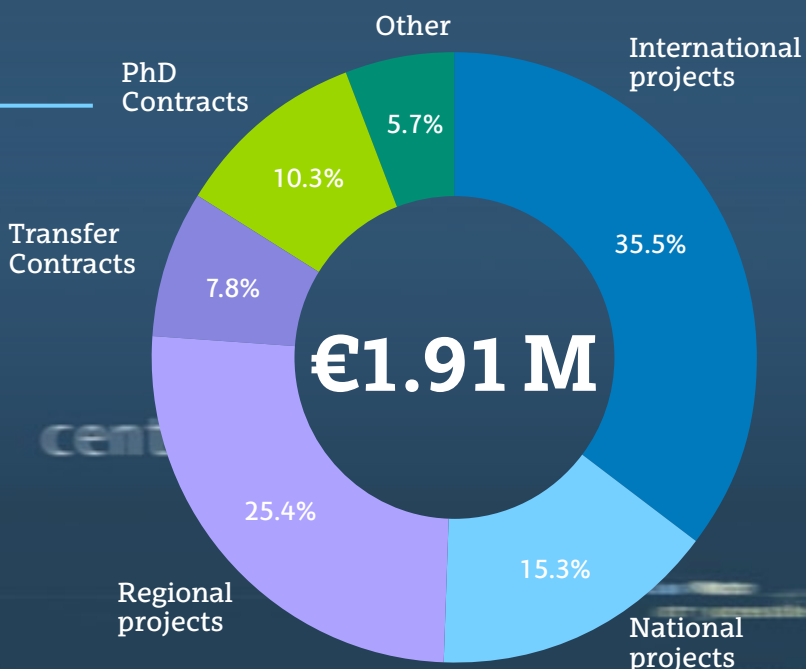


# People — 108

	Management and administration	3
	Lecturers	42
	PhD Grants	9
	Postdoctoral researchers	8
	Ramón y Cajal	1
	HAP/LAP students	9
	PhD students	34
	Doctoral theses defended	2



# Budget —



# ORGANIZATION

HiTZ is a multidisciplinary research center on **Language-centric Artificial Intelligence** with members from seven departments of the University of the Basque Country. The objective of the center is to **investigate** language and speech technologies, with a significant effort towards the **transfer** of knowledge and technology to companies. It comprises two research groups Aholab and Ixa, both with extensive experience since 1993, performing basic research, creating linguistic resources and tools and launching different commercial products on the market. HiTZ is also a **founding member** of **CLARIN-K Center**, member of **CLAIRE** and full member of **BDVA** and **DAIRO**. Through CLAIRE and BDVA, we also participate in the European Partnership on Artificial Intelligence, Data and **Robotics**.



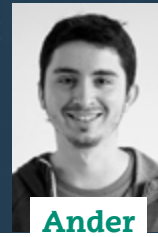
**Eneko Agirre**

Director



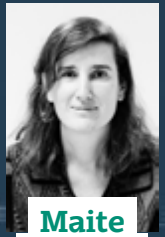
**German Rigau**

Subdirector



**Ander Salaberria**

Board member



**Maite Oronoz**

Board member

The members of the center are international referents in their scientific areas. At the moment, it is formed by **more than 60 members**, including computer scientists, linguists and 3 research technicians. In the last five years, the researchers now in the center have published more than 200 scientific publications. The group is a leader in applying deep learning techniques to language processing and its recent work in the area has been **cited more than 4,000 times** in the last two years.

The members of the center have been **advisors** in the creation of the National Plan for Spanish Language Technologies and are currently advising the Basque Government's equivalent counterpart.

**CLARIN  
K CENTRE**

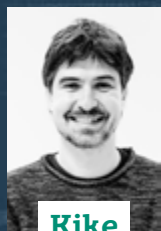


**CLAIRE**





# ahō LAB



**Kike  
Fernandez**

Board member



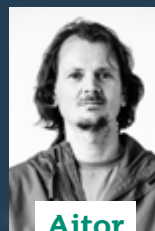
**Itziar  
Aldabe**

Board member



**Inma  
Hernaez**

Board member



**Aitor  
Soroa**

Board member


Both IXA and Aholab have been evaluated as high-performance research groups in the last research evaluation exercise by the science agency of the Basque Government. During their history, the groups have participated in more than 200 **research projects** ranging from regional to European projects. It has also participated in more than 100 **industrial contracts** with the aim of transferring technology into the industry.

HiTZ is also a member of **Erasmus Mundus+ European Masters Program** in Language and Communication Technologies (LCT) **program**. It is designed to meet the demands of industry and research in the rapidly growing field of

language technology. HiTZ also offers a **Doctoral Programme** in Language Analysis and Processing.

**The University of the Basque Country** (UPV/EHU) is the leading teaching and research institution in the Basque Country, a prosperous region stretching along the Atlantic coast of northern Spain. The UPV/EHU is among the best 400 universities in the world according to the Shanghai ranking, and has been recognized as an International Excellence Campus by the Spanish Government. The University of the Basque Country, a vibrant 30-year-old institution with 45,000 students, 5,000 world-class academic staff and state-of-the-art facilities distributed throughout 20 centers in its three campuses.

# RESEARCH AREAS

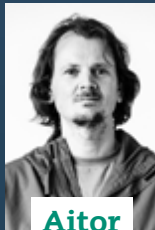
 Click on this symbol for more information



## Information Extraction and Information Retrieval



Main Researcher:



**Aitor Soroa**



## Machine Translation



Main Researcher:



**Gorika Labaka**



## Human-Computer Interaction



Main Researcher:



**Eneko Agirre**



## Speech and Language Resources



Main Researcher:



**Ainara Estarrona**

# INFRASTRUCTURE

15

multiprocessor GNU/Linux servers

4

SPARC Solaris servers

1

HPC Cluster with 128 cores

1

acoustically isolated room with audio equipment for professional recordings





### Text Analysis



Main Researcher:



**Rodrigo Agerri**



### Speech Technologies



Main Researcher:



**Inma Hernaez**



### Medical and Legal domains



Main Researcher:



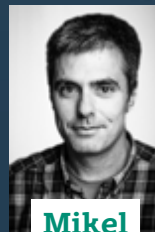
**Arantza Casillas**



### Digital humanities and education



Main Researcher:



**Mikel Iruskietia**

# hizkuntza

10 servers with

# 45

GPUs

Over

# 260 TB

of raw Network storage capacity

# 1 1

Behringer 4x4 audio/MIDI interface

Quiet PC Sentinel Fanless i10

# RESEARCH & TRANSFER



34

Research projects

11

Knowledge transference projects

5

Other projects

2

Doctoral theses defended

25

Journal papers (11 Q1)

36

Conference papers (8 A or A+)

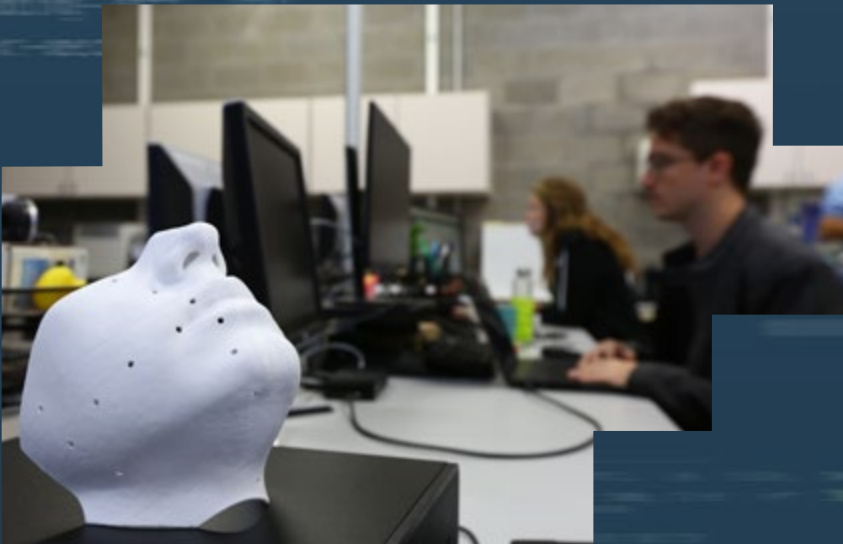
3

Book chapters

hitzak

10

BASQUE CENTER FOR LANGUAGE TECHNOLOGY



Language

words




**36**   
Students  
in masters

**34**   
Students  
in doctoral  
program

**TRAINING** 

**9**   
HAP/LAP  
Master Thesis  
Finalized

**75**   
Students in 2 Deep  
Learning complementary  
courses

**6**  
Ikasiker

**10**  
Internal and  
external  
internships

# ACTIVITIES



# 40

Seminars



# 12

Webinars

# 7

Workshops

12

# 7

Awards

Association of Computational Linguistics (ACL) Fellow

National Research Award in Computer Science

Young Researcher Award in Computer Science

SEPLN Prize for Best Doctoral Thesis in NLP

2020 EurAI Doctoral Dissertation Award

SCIE-Fundación BBVA Research Prize for Young Computer Science Researchers

Ikerkagazte 2021 Udabiltza Special Prize

